Naval Research Laboratory
Stennis Space Center, MS 39529-5004

NRL/FR/7322--95-9637

Software Design Document for the Polar Ice Prediction System Version 2.0

RUTH H. PRELLER
PAMELA G. POSEY

Ocean Dynamics and Prediction Branch
Oceanography Division

December 15, 1997

Approved for public release; distribution unlimited.

REPORT DOCUMENTATION PAGE

Form Approved
OBM No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 15, 1997	Final

4. TITLE AND SUBTITLE

Software Design Document for the Polar Ice Prediction System Version 2.0

5. FUNDING NUMBERS

Job Order No. 573121496
Program Element No. 0603207N
Project No.
Task No. X2003
Accession No. DN153095

6. AUTHOR(S)

Ruth H. Preller and Pamela G. Posey

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Research Laboratory
Oceanography Division
Stennis Space Center, MS 39529-5004

8. PERFORMING ORGANIZATION REPORT NUMBER

NRL/FR/7322--95-9637

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Space and Naval Warfare Systems Command
Washington, DC 20361

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Since 1987, the Fleet Numerical Meteorology and Oceanography Center (FNMOC) has been running sea ice forecasting systems in various regions of Navy interest (the Central Arctic, the Barents Sea, and the Greenland Sea). The Polar Ice Prediction System (PIPS1.1) predicts sea ice conditions in the Arctic basin, the Barents Sea, and the Greenland Sea at a resolution of 127 km. Two regional sea ice forecasting systems, the Polar Ice Prediction System – Barents Sea (RPIPS-B) and the Polar Ice Prediction System – Greenland Sea (RPIPS-G), also predict sea ice conditions in the Barents and Greenland Seas, respectively, at a higher resolution of 20–25 km. In 1995, the Naval Research Laboratory delivered to FNMOC a coupled ice-ocean system, the Polar Ice Prediction System 2.0 (PIPS2.0), which predicts sea ice conditions of most of the ice-covered regions in the Northern Hemisphere. PIPS2.0 will replace the three existing operational forecast systems when it completes the final operational testing phase at FNMOC. PIPS2.0 uses as its basis the Hibler ice model and the Cox ocean model. PIPS2.0 has a resolution of approximately a quarter of a degree, which is similar to the resolution of the operational regional systems (RPIPS-B and RPIPS-G). This report briefly describes the main program and each of its subroutines.

14. SUBJECT TERMS

ice forecast, ice drift, ice edge, ice model

15. NUMBER OF PAGES
151

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	Same as report

CONTENTS

SOFTWARE DESIGN DOCUMENT FOR THE POLAR ICE PREDICTION SYSTEM VERSION 2.0

1.0 INTRODUCTION

1.1 Scope

This Software Design Document (SDD) describes the Computer Software Configuration Item (CSCI) identified as the Polar Ice Prediction System 2.0 (PIPS2.0). This SDD has been prepared in accordance with guidelines set forth by the Fleet Numerical Meteorology and Oceanography Center (FNMOC). These guidelines are based on the Data Item Description (DID) DI-MCCR-80012A of DOD-STD-2167A.

1.2 System Overview

PIPS2.0 was developed as an ice-ocean coupled model to provide daily forecasts of ice-drift velocity, ice thickness, and ice concentration for most ice-covered regions in the Northern Hemisphere. (This includes the area from the North Pole south to approximately 30° N.)

Two independent models were merged to form PIPS2.0: the Hibler Viscous-Plastic Sea Ice Model, which provides the ice prediction output, and the Cox Ocean Model, which provides the ocean forcing required as input for the Hibler ice model. To accomplish the merger, the models were first independently adapted to the required prediction basin and then joined by a common driver routine. Information between the coupled models is exchanged via common blocks.

In this configuration, the ocean model provides daily predictions of mixed-layer temperatures, variable freezing temperatures, oceanic heat fluxes, and ocean currents to the ice model, while the ice model supplies the ocean model with ice concentration, ice growth rate, ice thickness, ice thickness growth rate, ice surface temperature, ice-drift velocity, and heat above the freezing temperature.

The basis of the ice model is the Hibler dynamic/thermodynamic ice model (Hibler 1979; 1980) that was modified for operation in the polar regions (Preller and Posey 1989) and updated for spherical coordinates (Cheng and Preller 1992). The ocean model is the Cox primitive equation, numerical model (Cox 1984) that predicts horizontal and vertical velocities, temperature, and salinity for a three-dimensional ocean basin. Daily Naval Operational Global Atmospheric Prediction System (NOGAPS) data supplies the atmospheric data for forcing.

The PIPS2.0 ice-ocean coupled system is presently in research and development at the Naval Research Laboratory (NRL), Stennis Space Center, MS, and is being implemented at FNMOC. The model is designed to run on a UNIX host platform. For greater applicability, host system specifics are kept to a minimum in this document.

1.3 Overview

This SDD provides a complete description of the design of CSCI PIPS2.0. It describes PIPS2.0 as composed of Computer Software Components (CSC) and Computer Software Units (CSU) and details the allocation of requirements from PIPS2.0 to its CSCs and CSUs. The overview of the design requirements of each PIPS2.0 CSC is contained in Sec. 2.0. Section 3.0 contains the detailed design information for each CSC and CSU. Global data elements are listed and described in Sec. 4.0, and Sec. 5.0 provides a data file to CSU cross reference. Abbreviations and acronyms used in this document are listed in Sec. 6.0.

2.0 GENERAL DESIGN

2.1 CSCI Overview

The PIPS2.0, is based on the Hibler sea ice model coupled with the Cox ocean model. PIPS2.0 takes the place of all operational versions of the ice model at FNMOC.

The Hibler ice model (1979; 1980) was originally developed as a stand-alone program to predict ice-drift velocity, ice thickness, and ice concentration on a daily basis. The Cox ocean model (1984) is coupled with the ice model to provide the ocean currents, temperature, and salinity values. The ocean model provides oceanic forcing, including ocean currents, temperature, salinity, and oceanic heat fluxes to the ice model. The ice model provides the ocean model with predictions of ice conditions including ice concentration, ice growth rate, ice thickness, ice thickness growth rate, ice surface temperature, ice-drift velocity, and heat above the freezing temperature. The two models exchange information via common blocks. Ice and ocean restart data are updated on a daily basis—output one day, input the next.

All the external interfaces to PIPS2.0 are shown in Fig. 2.1-1. The first three interfaces in the second row of input are all output from the previous day's run. Run-specific parameters are user input. Additional model disk file interfaces include subsets of the Levitus climatology data base, river discharge rates, and the NOGAPS data. Levitus climatology and river discharge rates vary monthly, while NOGAPS data vary daily. Land/sea masks and Earth-oriented latitudes and longitudes that correspond to the ocean basin grid cells (Sec. 2.2) are also input.

The following list is a summary of the external interfaces:

Ice Restart File

Previous Run – The ice conditions including ice drift, ice thickness, ice (or water) temperature and ice coverage are read in as restart data.

Current Run – The ice conditions including ice drift, ice thickness, ice (or water) temperature and ice coverage are output. This file becomes the restart file for the next PIPS2.0 run.

Ocean Restart File

Previous Run – The ocean conditions including currents, temperature, and salinity for the last two timesteps computed during the previous model run are read in as restart data.

Current Run – The ocean conditions of currents, temperature, and salinity computed for the last two timesteps during the current run are output. This file becomes the restart file for the next PIPS2.0 run.

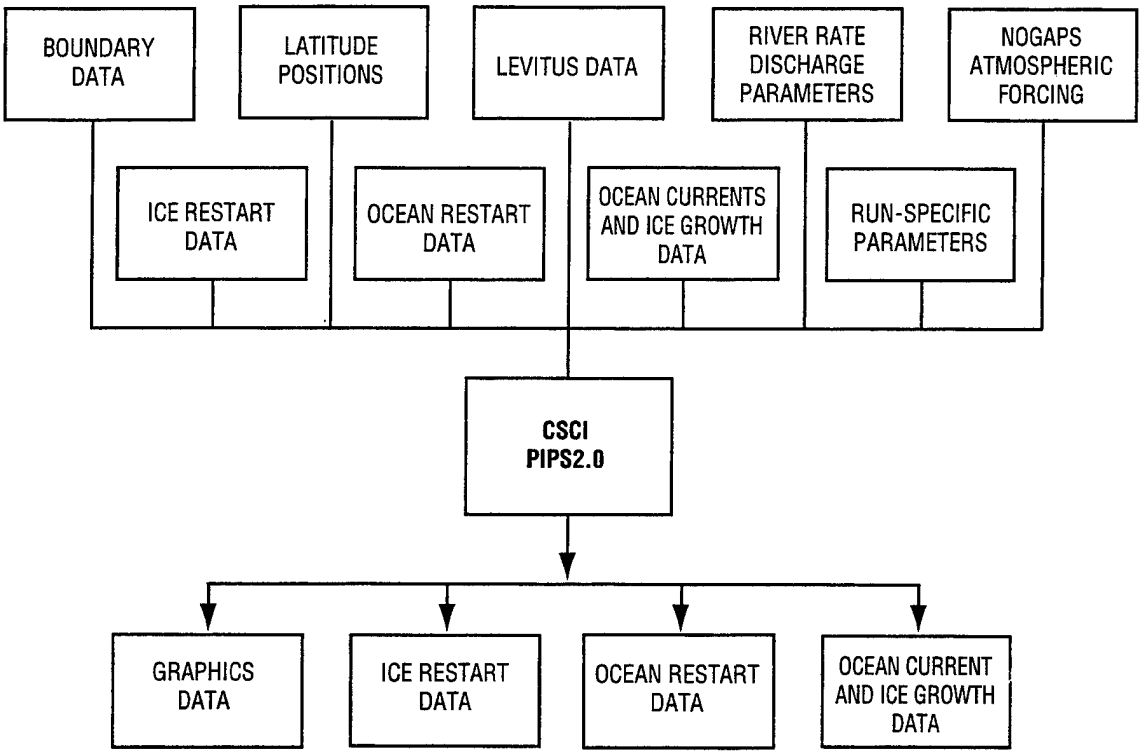

Fig. 2.1-1 — CSCI overview

Ocean Data – Ocean currents and ice growth rate data (including current, temperature, salinity, and ice growth rate) are used to initialize the timestepping.

Data File for Plotting – Fields of wind, ice thickness, ice drift, and ice concentration are output to an ASCII file for subsequent plotting.

Land/Sea Masking – The land/sea masks for thermodynamic fields, velocity files, and outflow grid cells are defined.

Latitude/Longitude in Earth-Oriented Spherical Coordinates – Grid positions provide Earth-oriented latitudes of U,V points needed for computing Coriolis force.

Monthly Levitus Temperature – Levitus ocean temperature data are interpolated to PIPS2.0 grid.

Monthly Levitus Salinity – Levitus ocean salinity data are interpolated to PIPS2.0 grid.

Monthly River Temperature Data – Monthly river discharge rates and temperatures are provided for 88 grid locations.

NOGAPS Atmospheric Forcing – NOGAPS daily atmospheric data include air temperature, surface forcing, surface vapor pressure, incoming solar radiation, total heat flux, and sensible heat flux to calculate geostrophic winds for forcing.

Run Specific Parameters – The number of timesteps for run, the interval at which to plot, the interval at which to print, the restart indicator, and the run date-time group are user input.

2.1.1 CSCI Architecture

The top-level architecture of PIPS2.0 is illustrated in Fig. 2.1-2. Each block of the figure represents a CSC described in Sec. 2.2.

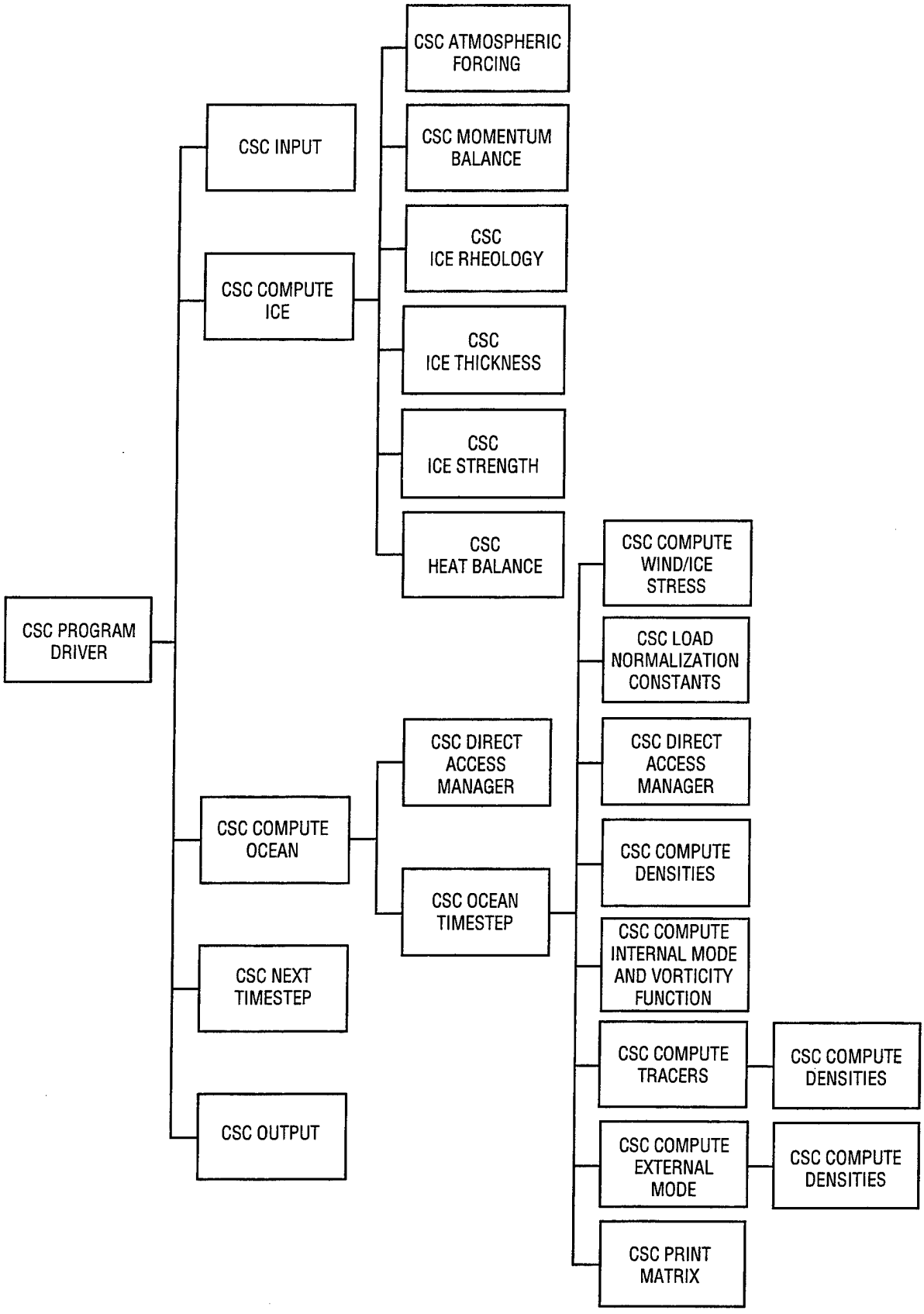

Fig. 2.1-2 — CSCI top-level architecture

Table 2.1-1 — CSCI Memory Requirements

Common Block	Number of Bytes	Common Block	Number of Bytes
BG	67,910,480	PRESS	518,400
CORSP	521,284	OCEANS	518,400
COX2	4,152,960	ONEDIM	568,156
CURNTS	1,555,200	RAD	521,284
DIFFU3	1,440	RFOR	2,606,420
FIELDS	2,073,600	RFOR2	2,073,616
FORCE	1,031,048	RSTRT	7,752,992
FULLWD	34,996	RVR	1,412
GROW	518,400	STEPSP	2,884
LEVITUS	15,638,520	TSTOP	16,594,568
MASK	1,552,324	WORKSP (R)	4,148,640
Total			130,298,436

2.1.2 System States and Modes

The ice model CSCI operates in only one state and in one mode: restart. In the restart mode, the inputs of ice drift, thickness, compactness, lateral heat, and ice temperature are read from the most recent model forecast, or from climatology, if the forecast restart is not available.

Provisions are made to allow for the future vectorization of parts of several CSCs including CSC Compute Internal Mode and Vorticity Driving Function, CSC Compute Tracers, and CSC Compute External Mode.

2.1.3 Memory and Processing Time

Most of the CSCI internal data is located in common blocks with only a relatively small amount of data designated as local to any particular CSC. For this reason, the memory requirements of each CSC are not addressed individually. The CSCI PIPS2.0 memory requirements are given in Table 2.1-1. The amount of local data contained in any of the CSCs is negligible in comparison to the total data stored in common areas. As an estimate, it is sufficient to say that the largest amount of memory capacity required for any CSC is about 2 Mb greater than the 130.3-Mb total given at the bottom of the table.

Because the external modes are computed after the row-by-row computations are performed by CSC Compute Internal Mode and Vorticity Driving Function, the common block WORKSP is redefined in CSC Compute External Mode. Therefore, the memory requirement given for common block WORKSP is the maximum of its two versions, which in this case is the WORKSP defined in CSC Compute External Mode. The common block WORKSP defined in CSC Compute External Mode is referred to as WORKSP (R) to distinguish it from the WORKSP common block used by the other CSCs. CSCs containing fewer than 100 bytes are not shown.

It is important to note that the memory requirements given here are for PIPS2.0 as it is currently configured as a coupling of the ocean and ice models. These memory requirements change significantly based on two factors:

1. If the program is not to be run in a core-contained mode, common block BG can be taken out of CSC Direct Access Manager, reducing the memory requirements by approximately 68 Mb. When the program is not run in a core-contained mode, data that would normally be resident in common block BG are output to disk files. The cost, however, is a substantial increase in execution time.
2. The majority of arrays are dimensioned according to parameters that define the number of grid cells contained in the defined model basin. Table 2-1.1 shows the amount of memory required to process the defined model basin. This model basin is comprised of 360 grid cells meridionally by 360 grid cells zonally by 15 grid cells vertically (or 1,944,000 grid cells). Because a multitude of data elements are associated with each grid cell, the amount of required memory varies significantly with the number of grid cells in the model basin.

2.2 CSCI Design Description

PIPS2.0 provides predictions for a model basin that extends from 30° N to the North Pole. The basin is divided into 359 × 359 grid cells based on a spherical coordinate system. Each grid cell has an arc length of 0.57°. To remove computational instabilities, the Earth-oriented coordinates are transformed into a new geometry. This is accomplished by first rotating the Earth's longitudes so that the Prime Meridian is located at 190° E. Then the North Pole is rotated 90° down the 100° E meridian until it resides on the true equator. In this transformed system, the coordinates have the 170° W to 10° E great circle, which passes through the North Pole as the "new" (repositioned) equator.

2.2.1 CSC Program Driver

This CSC serves as the main driving routine for PIPS2.0. It must establish the proper data in memory needed to restart the model each day, drive the ice and ocean CSCs, and save the data needed for the next model restart.

2.2.2 CSC Input

CSC Input is used for reading and/or calculating inputs necessary for the operation of PIPS2.0. It reads in and converts the date-time group to its components, reads in boundary mask data, and reads in the ice and ocean model restart data files from the previous PIPS2.0 run.

2.2.3 CSC Compute Ice

CSC Compute Ice must provide all daily ice data predictions required by FNMOC and must provide data required by CSC Compute Ocean. CSC Compute Ice must contain the capability to compute the following elements:

- ice-drift velocity
- ice thickness
- ice thickness growth rate
- ice concentration
- heat above the freezing temperature
- oceanic heat fluxes
- geostrophic winds

2.2.3.1 CSC Atmospheric Forcing

CSC Atmospheric Forcing reads the atmospheric forcing fields that are derived from the 1992 results of NOGAPS. Geostrophic winds are calculated from the input fields.

2.2.3.2 CSC Momentum Balance

The Momentum Balance CSC calculates the terms needed to solve the momentum equation. It estimates the x and y components of forcing due to the ocean currents plus the ice pressure gradient, which are then used to solve the momentum equation through the relaxation method.

2.2.3.3 CSC Ice Rheology

The Ice Rheology CSC calculates the ice stress that is directly related to the ice strength and strain rates. It calculates strain rates, divergence, and viscosities based on plastic flow specified by an elliptic yield curve.

2.2.3.4 CSC Ice Thickness Distribution

The Ice Thickness Distribution CSC accounts for the changes in ice thickness and concentration due to growth, advection, and deformation of the ice. Only thick and thin ice are considered for these calculations. The deformation of thick ice can create thin ice by divergence, while the thin ice can be removed by convergence. Both growth and melt affect the amount of thick and thin ice. While growth can significantly decrease the amount of thin ice and increase the amount of thick ice, melting can add to the amount of thin ice or create open water by decreasing the thick ice. The advection of the ice thickness and compactness due to explicit timestepping is performed. The diffusion of the ice thickness, compactness, and concentration is determined by using explicit forward time differencing. The negative ice to be melted is calculated, and the changes of thickness and compactness for each timestep are then estimated.

2.2.3.5 CSC Ice Strength

The ice strength, as a function of ice thickness distribution, is calculated in CSC Ice Strength. The strength of the ice depends on the amount of thin ice and is calculated in the FORM CSU.

2.2.3.6 CSC Heat Balance

The Heat Balance CSC takes the FNMOC forcing and calculates the terms needed for the heat budget and uses these terms to compute the growth rates of thick and thin ice. It also computes the surface temperature of ice by iteration. This temperature balances the surface heat budget and dictates the conduction of heat through the ice and, therefore, the growth rates.

2.2.4 CSC Compute Ocean

CSC Compute Ocean must provide the oceanic forcing required for CSC Compute Ice. This includes daily predictions of mixed-layer temperatures, variable freezing temperatures, oceanic heat fluxes, and ocean currents. The CSC Compute Ocean must be configurable so that predictions can be provided for a model basin that is compatible with any basin processed by CSC Compute Ice.

2.2.4.1 CSC Ocean Timestep

CSC Ocean Timestep must initialize all quantities needed to begin processing for each timestep and must control the row-by-row computation of prognostic variables for each timestep. Data for each timestep is processed in rows, with a row being a three-dimensional vertical slice or "slab" of the ocean basin comprised of i, j, k grid cells of constant index j with index i varying from western to eastern basin boundaries and with index k varying from the ocean surface to the bottom. (See Fig. 2.2-1 for the layout of the horizontal and vertical grid spacing and indexing.)

Quantities initialized by CSC Ocean Timestep before row-by-row computations can begin are those needed for analysis of the calculation including, for example, change of variance of tracers, rates of change of kinetic energy, and mass transport of tracers. For each row processed, CSC Ocean Timestep establishes the proper data in memory before passing control to CSC Compute Internal Mode and Vorticity Driving Function and CSC Compute Tracers.

2.2.4.1.1 CSC Compute Wind/Ice Stress – CSC Compute Wind/Ice Stress must compute daily wind stress at the sea ice surface and the stress between the sea ice and the mixed layer. This computation is required at the beginning of each timestep processed.

2.2.4.1.2 CSC Load Normalization Constants – CSC Load Normalization Constants loads constants and coefficients for use in CSC Compute Densities. Arrays must be filled with coefficients of the equation of state, and normalizing temperatures and salinities.

2.2.4.1.3 CSC Compute Densities – CSC Compute Densities must provide the normalized densities for all grid cells in the bootstrap row processed by CSC Ocean Timestep and for all rows processed by CSC Compute Tracers and CSC Compute Internal Mode and Vorticity Driving Function. The grid cell densities are required for purposes of checking vertical stability.

2.2.4.1.4 CSC Print Matrix – CSC Print Matrix will print two-dimensional arrays as required by CSC Ocean Timestep. It must accept variable size arrays and must print contiguous array elements as specified by CSC Ocean Timestep.

2.2.4.2 CSC Compute Internal Mode and Vorticity Driving Function

For each row processed by CSC Ocean Timestep, CSC Compute Internal Mode and Vorticity Driving Function computes the internal mode components of the U and V velocities and the vorticity driving function for use in computing the external mode velocity. The terms U and V represent the zonal and meridional components, respectively, of velocity. In the horizontal plane, U and V are situated at the grid cell corners. In the vertical plane, U and V are located halfway through the vertical dimension of the grid cell.

CSC Compute Internal Mode and Vorticity Driving Function must compute the factors that contribute to the internal mode component of velocity including total advection of momentum, horizontal and vertical diffusion of momentum, Coriolis force, and hydrostatic pressure components. The following equations, given in simplified terms, are used to calculate velocity internal mode components:

$$UA_{tau} = UA_{tau-1} + 2 * DTUV * (UAM + UHD + UVD + UCF + UHP) \quad (1)$$

$$VA_{tau} = VA_{tau-1} + 2 * DTUV * (VAM + VHD + VVD + VCF + VHP) \quad (2)$$

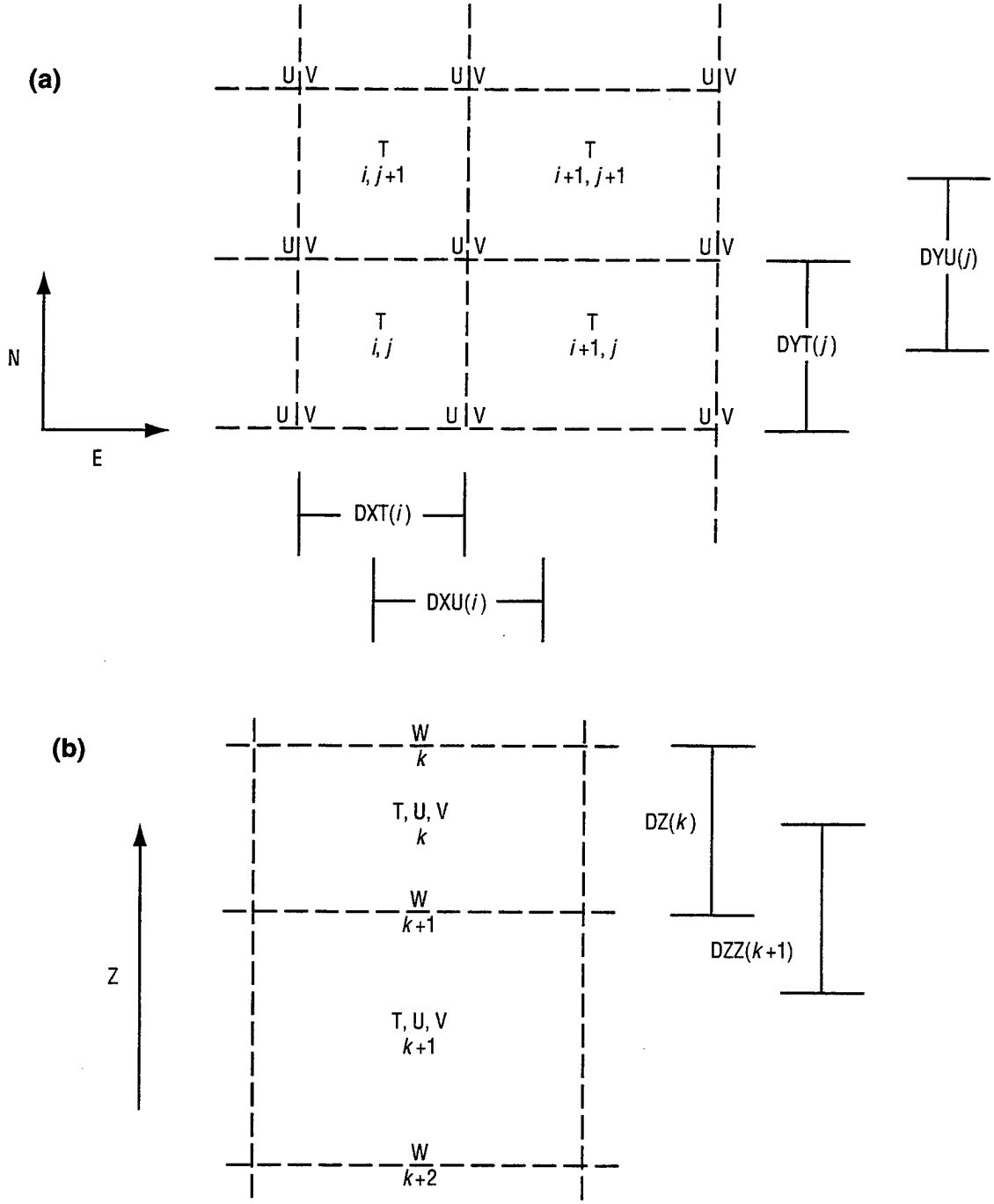

Fig. 2.2-1 — (a) Horizontal and (b) vertical grid spacing and indexing

where

UA_{tau}, UA_{tau-1}, VA_{tau}, and VA_{tau-1} represent zonal and meridional internal mode components of horizontal velocity for present and previous timesteps,

DTUV is the length in seconds of the timestep on U,V,

UAM and *VAM* are zonal and meridional components of total advection of momentum,

UHD and *VHD* are zonal and meridional components of horizontal diffusion of momentum,

UVD and *VVD* are zonal and meridional components of vertical diffusion of momentum,

UCF and *VCF* are zonal and meridional components of Coriolis force, and

UHP and *VHP* are zonal and meridional components of hydrostatic pressure.

2.2.4.3 CSC Compute Tracers

CSC Compute Tracers will compute ocean model tracer elements, including temperature and salinity, for each row processed by CSC Ocean Timestep. Tracer elements represented by T are located at the horizontal center of the ocean model grid cells and halfway through the vertical dimension.

CSC Compute Tracers must compute components of the tracer equation including total advection of tracers, horizontal and vertical diffusion of tracers, and surface boundary conditions. The total advection of the tracers is equal to the sum of the flux through the west face of the T box, plus the flux through the top of the T box, plus the zonal, meridional, and vertical flux divergences.

Implementation of the following equation is required for computing water temperature:

$$\frac{\partial T}{\partial t} + \nabla \bullet (uT) = K_H \nabla_H^2 T + K_Z \frac{\partial^2 T}{\partial z^2} - \frac{f(0)(1-A)\,\delta(z) R_0 \theta \left(T - T_f\right)}{Z_{mix}} - R_t \left(T - T_0\right), \qquad (3)$$

where

$\frac{\partial T}{\partial t}$ = the time rate of change of temperature,

$\nabla \bullet (uT)$ = heat advected into the T box,

$K_H \nabla_H^2 T$ = horizontal diffusion,

$K_Z \frac{\partial^2 T}{\partial z^2}$ = vertical diffusion,

$f(0)$ = the ice growth/melting rate in open water,

$(1 - A)$ = the percentage of the open water area,

$\delta(z)$ = 1 in the mixed layer and 0 otherwise,

R_0 = the ratio of the latent heat of fusion of sea ice to the heat capacity of water,

$\theta(T - T_f)$ = 1 when the mixed-layer temperature T is greater than the freezing point T_f and 0 otherwise,

Z_{mix} = the mixed-layer thickness,

R_t = the robust constraint for water temperature, and

T_0 = the Levitus climatology temperature.

Similarly, implementation of the following equation is required for computing salinity:

$$\frac{\partial S}{\partial t} + \nabla \bullet (uS) = K_H \nabla_H^2 S + K_Z \frac{\partial^2 S}{\partial z^2} - \frac{0.035 G_h \delta(z)}{Z_{mix}} - R_t \left(S - S_0\right), \quad (4)$$

where

$\frac{\partial S}{\partial t}$ = the time rate of change of salinity,

$\nabla \bullet (uS)$ = salinity advected into the T box,

$K_H \nabla_H^2 S$ = horizontal diffusion,

$K_Z \frac{\partial^2 S}{\partial z^2}$ = vertical diffusion,

G_h = the total growth rate of open water and sea ice,

$\delta(z)$ = 1 in the mixed layer and 0 otherwise,

Z_{mix} = the mixed-layer thickness,

R_t = the robust constraint for water temperature, and

S_0 = the Levitus climatological salinity.

2.2.4.4 CSC Compute External Mode

CSC Compute External Mode must employ sequential overrelaxation to solve the LaPlacian equation for the external mode of velocity in terms of a mass transport stream function. To provide predictions for a model basin with islands, CSC Compute External Mode must utilize the method of "hole relaxation." The vorticity driving function computed by CSC Compute Internal Mode and Vorticity Driving Function is required as input to CSC Compute External Mode.

CSC Compute External Mode must read in the relaxation solutions from the previous two timesteps and extrapolate the solutions forward in time for use as an initial guess for the relaxation solution of the current timestep. This minimizes the number of scans required for convergence to the solution.

2.2.4.5 CSC Direct Access Manager

CSC Direct Access Manager is included in ocean to perform input/output (I/O) tasks on slab and slab incidental data. The I/O may be real, when data is actually stored on disk, or simulated, when all data is resident in memory (core-contained). Either way, CSC Direct Access Manager insulates the rest of the model from any dependence on the type of I/O system actually used. When the ocean model is configured to run in a core-contained mode, CSC Direct Access Manager contains routines to initialize a virtual disk from tape, transfer data to memory and vice versa, and save a virtual disk to tape.

The I/O system must satisfy two main criteria. First, a complete record of all prognostic variables must reside on permanent disk at the end of each timestep. This allows restarting the model for continuous daily execution. Secondly, the I/O system must feed data to and from memory in a row-by-row manner. Data for one row including all east-west and vertical gridpoints is termed slab data. At any one time, only the slab data necessary for the computation should be present. This includes data for the previous, the current, and the next timestep.

Three disk units are needed for the I/O system: one for the N timestep data, one for the $N-1$ timestep data, and one for the newly computed $N+1$ timestep data. Disk units are permuted between units 13, 14, and 15 to minimize data transfers. Primary slab data including tracer data and U and V velocities are managed as follows: On timestep 1, $N-1$ and N data will be read from units 13 and 14, and the $N+1$ data will be written to 15. On timestep 2, units 14 and 15 will be read and the $N+1$ data will be written to unit 13. On timestep 3, 15 and 13 will be read and the $N+1$ data will go to 14, etc. In a core-contained mode, the file I/O described above is simulated by CSC Direct Access Manager.

CSC Direct Access Manager also controls the I/O for the slab incidental data, such as the number of vertical levels of the ocean at T and U,V points. Because these variables are constant in time, there is no need to keep multiple records of them. Slab incidental data are managed in a manner similar to the primary slab data with some considerations. The method is outlined in Sec. 3.4.6. CSC Direct Access Manager also manages the storage and retrieval of the two-dimensional horizontal fields and the timestep counter, total elapsed time, and the area and volume of the basin.

2.2.5 CSC Next Timestep

CSC Next Timestep is used after each timestep run of CSCs Ice and Ocean to update timestep information. It calculates new timestep information including date-time group, new currents, and new mixed-layer temperatures for the next pass through the ice and ocean models. Positions of previous and present fields are swapped.

2.2.6 CSC Output

The CSC Output is the last CSC executed in PIPS2.0. It is responsible for writing the data files containing the ice model and ocean model restart data. These restart values are used by PIPS2.0 for the next model run.

3.0 DETAILED DESIGN

This section gives design information for each PIPS2.0 CSC described in Sec. 2.0. Each CSC is divided into CSUs.

3.1 CSC Program Driver

CSC Program Driver serves as the main driving routine for PIPS2.0. It must establish the proper data in memory needed to restart the model each day, drive the Ice and Ocean CSCs, and save the data needed for the next model run.

3.1.1 CSU DRIVER

CSU DRIVER serves as the program driver for CSCI PIPS2.0.

3.1.1.1 CSU DRIVER Design Specification/Constraints

There are no known constraints.

3.1.1.2 CSU DRIVER Design

Input Data Elements:

FCORSP	array of sine of latitude positions for each gridpoint
IDTG	date-time group in the form YYMMDDHH
ITSTEP	number of timesteps for run
PLTSTP	interval in timesteps at which to plot
PRTSTP	interval in timesteps at which to print
IRIVER	x component of river discharge
JRIVER	y component of river discharge
KRIVER	z component of river discharge
RIVER	river discharge temperature
SMIX	monthly interpolated Levitus salinity fields
TMIX	monthly interpolated Levitus temperature fields

Output Data Elements:

GAIRX	x component of geostrophic wind
GAIRY	y component of geostrophic wind
HEFF	mean ice thickness per grid cell
UICE	x component of ice drift
VICE	y component of ice drift
AREA1	fraction of grid cell covered with ice

/FCORSP/

FCORSP	array of sine of latitude positions for each gridpoint

/FULLWD/

NNERGY	
NLAST	final timestep to compute for the current run

/LEVITUS/

SMIX	monthly interpolated Levitus salinity fields
TMIX	monthly interpolated Levitus temperature fields

/MASK/

HEFFM	land/sea mask for thermodynamic variables
OUT	land/sea mask including outflow conditions for thermodynamic variables
UVM	land/sea mask for velocity variables

/RFOR 2/

GWATX	x component of the ocean current
GWATY	y component of the ocean current

/RSTRT/

AREA1	fraction of grid cell covered with ice
HEFF	mean ice thickness per grid cell
TICE	mixed-layer temperature if open water; ice temperature if ice cover
UICEC	intermediate x component of ice drift
VICEC	intermediate y component of ice drift
UICE	x component of ice drift
VICE	y component of ice drift

/RVR/

IRIVER	x component of river discharge
JRIVER	y component of river discharge
KRIVER	z component of river discharge
RIVER	river discharge temperature

/TSTEP/

IDTG	date-time group in the form YYMMDDHH
IRSTRT	if 0, restart from previous run; otherwise, restart from constant conditions
ITSTEP	number of timesteps for run
MDY	day calculated from inputted data
MHR	hour calculated from inputted data
MM	month calculated from inputted data
MYR	year calculated from inputted data
PLTSTP	interval in timesteps at which to write output
PRTSTP	interval in timesteps at which to plot output

Parameters:
stored in file ice.par

IMTM1	IMT-1; = 359
JMTM1	JMT-1; = 359
IMT	total number of T grid boxes zonally; = 360
JMT	total number of T grid boxes meridionally; = 360
IMTP1	IMT+1; = 361
JMTP1	JMT+1; = 361
NXM1	IMT-2; = 358
NYM1	JMT-2; = 358
MIDY	80
IRV	80
LSEG	maximum number of sets of start and end indices; = 10
NISLE	number of islands in model; = 4
NT	number of tracer type variables carried in the model; = 2
KM	total number of vertical levels; = 15

Data element design information is provided in Sec. 4.0

Local Data Elements:

I	index counter	integer
IJUL	Julian date	integer
ILA	index of loop over rows/latitudes	integer
ILO	index of loop over columns/longitude	integer
J	index counter	integer

K	index counter	integer
MM1	index for the Julian date of the previous month	integer
NRV	index counter	integer
RADIAN	degrees per radian	integer

Logic Flow:

Upon start of the program, several inputs are read from standard input. These inputs include the number of timesteps for the run (ITSTEP), the interval in timesteps at which to plot the output (PLTSTP), the interval in timesteps at which to print the output (PRTSTP), the type of initialization (IRSTRT), and the date-time group (IDTG). The DAYNUM CSU is called to compute the year (MYR), month (MM), day (MDY), and hour (MHR) from the input date-time group. From the month, day, and hour, the final timestep to compute for the current run is calculated.

The BNDRY CSU is called to read the velocity, thermodynamic, and outflow land/sea boundary parameters from a file. The latitude positions of the gridpoints are read from a file and are then converted from degrees to radians. The values of IRIVER, JRIVER, KRIVER, and RIVER are also read from a file for every IRV. The ice and ocean restart values are read by calling the RST_OCN and RST_ICE CSUs. The Levitus temperature and salinity fields (TMIX and SMIX) are read from input files.

After all input data are read, the model is executed by running the CSC Ice and CSC Ocean portions of PIPS2.0 for each timestep. For each timestep, CSC Next Timestep is called to update the new timestep, moving previous timestep fields back one timestep and moving present timestep fields into previous timestep fields.

Once all timesteps have been executed, the ice and ocean model restart data are written to files by calling the SAV_ICE and SAV_OCN CSUs. A graphics file of output fields is also written directly from CSU DRIVER. These values include the x component of the geostrophic wind (GAIRX), the y component of the geostrophic wind (GAIRY), the mean ice thickness per grid cell (HEFF), the x component of the ice drift (UICE), y component of the ice drift (VICE), and the fraction of grid cell covered with ice (AREA1).

Local Data Files:

The CSU Driver reads data from five different data files. The file connected to logical unit 14 is named newlatu.dat and is a permanent, IEEE binary file containing the latitude positions of gridpoints. The river rate discharge data is in the file named river_*xxx*.dat where *xxx* is the month; this file is a permanent, ASCII file connected to logical unit 19. The files named for018_tu_*xx*.dat and for018_su_*xx*.dat, where *xx* is the month number, are connected to logical units 10 and 11, respectively. Both files are permanent, IEEE binary files. The "tu" file contains the interpolated Levitus monthly ocean temperature and the "su" file contains the interpolated Levitus monthly ocean salinity values. The final file contains the ice model results and is named 92*xxxx*.dat where *xxxx* is the month and day. This file is also a permanent, binary file and is connected to logical unit 31.

3.2 CSC Input

CSC Input is used for reading and/or calculating inputs necessary for operation of PIPS2.0. Four CSUs are called to perform the functions required by the CSC Input.

DAYNUM	converts date-time group
BNDRY	reads the boundary data

RST_ICE	reads ice model restart data written in the previous run of the model
RST_OCN	reads ocean model restart data written in the previous run of the model

3.2.1 CSU DAYNUM

The DAYNUM CSU converts the date-time group in the form YYMMDDHH to the year, month, day, and hour.

3.2.1.1 CSU DAYNUM Design Specification/Constraints

There are no known constraints.

3.2.1.2 CSU DAYNUM Design

Input Data Elements:

/TSTEP/

IDTG	date-time group in the form YYMMDDHH

Output Data Elements:

/TSTEP/

MDY	day from the date-time group
MHR	hour from the date-time group
MM	month from the date-time group
MYR	year from the date-time group

Data element design information is provided in Sec. 4.0

Local Data Elements:

IDTG1	temporary variable used in determining the day	integer
IDTG2	temporary variable used in determining the hour	integer

Algorithms:

To calculate the year (*MYR*), day (*MDY*), and hour (*MHR*), use the following:

$MYR = IDTG/1000000$

$MM = (IDTG - (MYR * 1000000))/10000$

$IDTG1 = (IDTG/10000) * 10000$

$MDY = (IDTG - IDTG1)/100$

$IDTG2 = (IDTG/100) * 100$

$MHR = IDTG - IDTG2$

where *IDTG* is the date-time group in the form YYMMDDHH. The date-time group IDTG is input to the CSU and the year, day, and hour are output.

Logic Flow:

The DAYNUM CSU is called by the ICEMDL CSU after it reads the date-time group and also after it increments the date-time group for the next timestep. The year, month, day, and hour are

calculated from the date-time group and the values are returned to the calling CSU. The month is then used in determining the snowfall rate.

3.2.2 CSU BNDRY

CSU BNDRY reads the velocity, thermodynamic, and outflow land/sea boundary parameters from a file.

3.2.2.1 CSU BNDRY Design Specification/Constraints

Land/sea boundary parameters are read from a file connected to logical unit 15.

3.2.2.2 CSU BNDRY Design

Input Data Elements:

UVM	land/sea mask for velocity variables
OUT	land/sea mask including outflow conditions for thermodynamic variables
HEFFM	land/sea mask for thermodynamic variables

Output Data Elements:
/MASK/

UVM	land/sea mask for velocity variables
OUT	land/sea mask including outflow conditions for thermodynamic variables
HEFFM	land/sea mask for thermodynamic variables

Parameters:
stored in file ice.par

The ice.par parameters are described in Sec. 3.1.1.2.

Data element design information is provided in Sec. 4.0.

Logic Flow:

The BNDRY CSU is called by CSU DRIVER to define all boundaries. The land/sea masks for the maximum number of velocity fields, thermodynamic fields, and outflow fields are read from a file connected to logical unit 15.

The land/sea masks for the thermodynamic, velocity, and outflow fields are returned to the CSU DRIVER.

Local Data Files:

The file connected to logical unit 15 is named mask_u.dat and is a permanent, binary file containing the land/sea tables that define the PIPS2.0 model. The file contains the land/sea masks for the thermodynamic fields, velocity fields, and outflow grid cells.

3.2.3 CSU RST_ICE

The RST_ICE CSU reads the ice restart data written from the previous ice model run.

3.2.3.1 CSU RST_ICE Design Specification/Constraints

Restart values are read from a file connected to logical unit 16.

3.2.3.2 CSU RST_ICE Design

Input/Output Data Elements:

AREA1	fraction of the grid cell covered by ice
HEFF	mean ice thickness per grid cell
TICE	mixed-layer temperature for open water or ice temperature for ice cover
UICE	x component of the ice drift
UICEC	intermediate x component of the ice drift
VICE	y component of the ice drift
VICEC	intermediate y component of the ice drift

Parameters:
stored in file ice.par

The ice.par parameters are described in Sec. 3.1.1.2.

Data element design information is provided in Sec. 4.0.

Logic Flow:

The x and y components of the ice drift (UICE and VICE), intermediate x and y components of the ice drift (UICEC and VICEC), mean ice thickness (HEFF), fraction of the grid cell covered by ice (AREA1), and mixed-layer temperature for open water or ice temperature for ice cover (TICE) are read from the permanent binary restart data file. All parameters are returned to the calling CSU.

This CSU is implemented by the CSU DRIVER.

Local Data Files:

The restart data are read from 92<*mmdd*>.res (unit 16), where <*mmdd*> is the two-digit month, and two-digit day of the previous day. This permanent, binary file contains the unformatted restart data from the previous day. The x and y components of the ice drift, the intermediate x and y components of the ice drift used in semi-implicit timestepping, the mean ice thickness per grid cell, the fraction of the grid cell covered by thick ice, the negative ice to be melted, and the mixed-layer temperature (in the case of open water) or the ice temperature (in the case of an ice cover) are used by the ocean model run for the current day and the ice model run for the next day.

3.2.4 CSU RST_OCN

The RST_OCN CSU reads the ocean model restart data written from the previous PIPS2.0 model run.

3.2.4.1 CSU RST_OCN Design Specification/Constraints

Restart values are read from a file connected to logical unit 13.

3.2.4.2 CSU RST_OCN Design

Input Data Elements:

FW1	heat above the freezing temperature
GICE	ice growth rate computed by the ice model
TM1	previous timestep mixed-layer temperature
TM2	previous two timesteps mixed-layer temperature

TMP	array containing previous timestep temperature and salinity values
UTEMP	previous timestep x component of ocean current
VTEMP	previous timestep y component of ocean current

Output Data Elements:

GWATX	x component of geostrophic ocean current
GWATY	y component of geostrophic ocean current

/COX2/

FW1	heat above the freezing temperature
GICE	ice growth rate computed by the ice model
SHICE	total ice thickness
TFRZ	temperature at freezing point
TM1	previous timestep mixed-layer temperature
TM2	previous two timesteps mixed-layer temperature
TMP	array containing previous timestep temperature and salinity values
UTEMP	previous timestep x component of ocean current
VTEMP	previous timestep y component of ocean current

/CURNTS/

T1	previous timestep of temperature and salinity
T2	previous two timesteps of temperature and salinity
U1	previous timestep of x component of ocean current
V1	previous timestep of y component of ocean current

/FULLWD/

ITT	timestep counter

/OCEANS/

FW	oceanic heat flux

Parameters:
stored in file ice.par

The ice.par parameters are described in Sec. 3.1.1.2.

Data element design information is provided in Sec. 4.0.

Local Data Elements:

CP	constant of water heat capacity	real
EXCHNG	number of times to exchange ice/ocean data	real
I	index counter	integer
J	index counter	integer
J1	temporary variable	integer
M	index counter	integer
RLATNT	constant of water latent heat	real
TDAY	timestep to exchange ice/ocean data	real
ZMIX	depth of mixed layer (30 m)	real

Logic Flow:

The x and y components of the currents, temperature, and the salinity from the ocean model are read from the permanent binary restart data file. Input currents are in centimeters per second,

temperatures are in degrees Celsius, and salinities are calculated from 0.035 – salinity to find the difference. Currents are converted to m/s and mixed-layer temperatures are converted to Kelvins. The seawater freezing temperature is calculated according to salinity as $-54.4 \times$ salinity. The u and v components of currents, ocean temperatures, and salinity for level 1 from the last two timesteps are stored.

The ice thickness growth rate of open water, the total ice thickness growth rate, and the heat above the freezing temperature from the previous timestep are read from the file. If the growth rate is greater than that of the heat above freezing temperature, and the heat above freezing is >0, the growth rate is set to 0 (no more cooling). Lastly, the heat flux is calculated.

Algorithms:

The oceanic heat flux is calculated using:

$$Q = c_p \; z_m \; (T_{m_{i-1}})/t_d/n_t + GL_w \,, \tag{5}$$

where c_p is the constant of water heat capacity,

z_m is the depth of the mixed layer,

T_{m_i} is the mixed-layer temperature from the previous timestep,

$T_{m_{i-1}}$ is the mixed-layer temperature from two timesteps ago,

t_d/n_t is the timestep to exchange data on (t_d is the number of seconds in a day and n_t is the number of times to exchange data),

G is the ice growth rate, and

L_w is the constant of water latent heat.

CSURST_OCN is implemented by CSU DRIVER.

Local Data Files:

The restart data are read from for010_<*mmdd*>.dat (unit 13), where <*mmdd*> is the two-digit month and two-digit day of the previous day. This permanent, binary file contains the unformatted restart data from the previous day.

3.3 CSC Compute Ice

3.3.1 CSC Compute Ice Driver

3.3.1.1 CSU ICEMDL

CSU ICEMDL is the main driving subroutine for the Hibler viscous-plastic sea ice model (1979; 1980). Minor modifications were made to allow the original routine to function as a subroutine in the CSCI PIPS2.0.

3.3.1.1.1 CSU ICEMDL Design Specification/Constraints – The snowfall rates are based on monthly climatological values (Maykut and Untersteiner 1969; Parkinson and Washington 1979). There are no known constraints.

3.3.1.1.2 CSU ICEMDL Design

Input Data Elements:

AMASS	ice mass per grid area
DELTAT	timestep (s)

DELTAX	x (longitude) grid spacing (deg)
DELTAY	y (latitude) grid spacing (deg)
DRAGA	asymmetric water drag plus the Coriolis parameter
DRAGS	symmetric water drag
ETA	nonlinear shear viscosity
FO	growth rate of thin ice
GAREA	change of compactness due to freezing or melting
HCORR	additional ice to be melted for mixed-layer balance
HDIFF1	net rate of total open water growth
ZETA	nonlinear bulk viscosity

/GROW/

FHEFF	total growth rate of thick ice

/MASK/

HEFFM	land/sea mask for thermodynamic variables
OUT	land/sea mask including outflow conditions for thermodynamic variables
UVM	land/sea mask for velocity variables

/RFOR2/

GAIRX	x component of the geostrophic wind
GAIRY	y component of the geostrophic wind
GWATX	x component of the ocean current
GWATY	y component of the ocean current

/RSTRT/

AREA1	fraction of grid cell covered by ice
HEFF	mean ice thickness per grid cell
UICE	x component of ice drift
UICEC	intermediate x component of the ice drift
VICE	y component of ice drift
VICEC	intermediate y component of the ice drift

/TSTEP/

IDTG	date-time group; read from standard input (YYMMDDHH)
ITSTEP	number of timesteps for run
MDY	day calculated from date-time group
MM	month calculated from date-time group
PRTSTP	interval in timesteps at which to write results

Output Data Elements:

A22	minimal compactness allowed
AMASS	ice mass per grid area
DIFF1	harmonic diffusion constant
DRAGA	asymmetric water drag plus the Coriolis parameter
DRAGS	symmetric water drag
ERROR	maximum error allowed in the relaxation scheme
ETA	nonlinear shear viscosity
FO	growth rate of thin ice
GAREA	change of compactness due to freezing or melting
HCORR	additional ice to be melted for mixed-layer balance

HDIFF1	net rate of total open water growth
HO	demarcation between thick and thin ice
LAD	determines the timestepping scheme used
THETA	indicates a backwards timestep
ZETA	nonlinear bulk viscosity

/GROW/

FHEFF	total growth rate of thick ice

/RFOR2/

GWATX	x component of the ocean current
GWATY	y component of the ocean current

/RSTRT/

AREA1	fraction of grid cell covered by ice
HEFF	mean ice thickness per grid cell
UICE	x component of ice drift
UICEC	intermediate x component of ice drift
VICE	y component of ice drift
VICEC	intermediate y component of ice drift

/SNOW/

SNRT	snowfall rate

/STEP/

DELTAT	timestep (s)
DELTAX	x (longitude) grid spacing (deg)
DELTAY	y (latitude) grid spacing (deg)

/STEPSP/

DELTXA	x (longitude) grid spacing for thermodynamic fields (m)
DELTXU	x (longitude) grid spacing for velocity fields (m)
DELTYA	y (latitude) grid spacing for thermodynamic fields (m)
DELTYU	y (latitude) grid spacing for velocity fields (m)

Parameters:
stored in file ice.par

The ice.par parameters are described in Sec. 3.1.1.2.

Data element design information is provided in Sec. 4.0.

Local Data Elements:

ARSUM	sum of ice compactness over all grid squares	real
ARSUM1	net ice concentration	real
DELTT	1/2 of the timestep (s)	real
FHSUM	sum of growth rate of ice over all grid squares	real
FHSUM1	net ice growth	real
GRSUM	sum of thin ice over all grid squares	real
GRSUM1	net open water growth	real
I	index counter	integer
ICOUNT	timestep counter	integer

IMT	total number of T grid boxes zonally (360)	integer
IMTM1	total number of T grid boxes zonally – 1 (359)	integer
IT	number of timesteps allowed during run	integer
ITSTEP	number of timesteps for run	integer
J	index counter	integer
JMT	total number of T grid boxes meridionally (360)	integer
JMTM1	total number of T grid boxes meridionally – 1 (359)	integer
KSTEP	counter for timesteps	integer
MIDY	(160)	integer
PRTSTP	the interval, in timesteps, at which to print results	real
RADIAN	degrees per radian (57.29578°)	real
RADIUS	radius of the earth (6370.0×10^3 m)	real
SM	maximum value of SMU and SMV	real
SMU	maximum value of x component of the velocity difference between times T+1 and T	real
SMV	maximum value of y component of the velocity difference between times T+1 and T	real
SQ	squared velocity	real
SQ1	squared velocity difference between times t+1 and t	real
THEFF	total basin ice thickness	real
THEFF1	total basin ice thickness after outflow adjustment	real
THEFF2	sum of thickness over all grid squares	real
TOUT	net outflow	real
TOUT1	outflow for the current timestep	real
UERR	x component velocity difference between time t+1 and t	real
VERR	y component velocity difference between time t+1 and t	real

Data Conversions:

When calculating the grid spacing in meters from the grid spacing in degrees, degrees must be converted to radians.

Logic Flow:

Before solving any equations, initial conditions for the model are defined. First, the latitude grid spacing in meters for the velocity (DELTYU) and thermodynamic (DELTYA) fields are defined. These values are assumed to be equal and are constant for all latitudes:

DELTYU = DELTAY * RADIUS/RADIAN

DELTYA = DELTAY * RADIUS/RADIAN,

where DELTAY is the y (latitude) grid spacing in degrees (0.28575°), RADIUS is the radius of the Earth (6370.0×10^3 m), and RADIAN is degrees per radian (57.29578°).

Next, the grid length in meters for the ice-drift velocity (DELTXU) and the grid length for ice thickness and concentration (DELTXA) are defined:

DELTXU(*J*) = cos(DELTAY * (FLOAT(*J*) – (MIDY – 0.5) – 0.5)/RADIAN)
* DELTAX * RADIUS/RADIAN

DELTXA(*J*) = cos(DELTAY * (FLOAT(*J*) – MIDY – 0.5)/RADIAN)
* DELTAX * RADIUS/RADIAN

where DELTAY is the y (latitude) grid spacing in degrees, J is the index of the number of T grid boxes meridionally and DELTAX is the x (longitude) grid spacing in degrees (0.28575°), RADIUS is the radius of the Earth, and RADIAN is degrees per radian.

The month and day are used to initialize the snowfall rate based on monthly climatological values (Parkinson and Washington 1979).

Month	**Snowfall Rate (m/s)**
January	3.215×10^{-9}
February	3.215×10^{-9}
March	3.215×10^{-9}
April	3.215×10^{-9}
May	19.29×10^{-9}
June	0
July	0
August 1–19	0
August 20–31	49.603×10^{-9}
September	49.603×10^{-9}
October	49.603×10^{-9}
November	3.215×10^{-9}
December	3.215×10^{-9}

Additional initial values are calculated. The maximum error allowed in the relation scheme (ERROR) is set to:

ERROR = 0.000001 * 5.0.

DIFF1, the harmonic diffusion constant, is redefined in spherical coordinates. It is assumed to be a constant coefficient for the harmonic function of the diffusion for ice concentration and thickness at all locations. Note that the harmonic term coefficient is defined to be 0.004 times the grid length to have the diffusion approximate to the change of ice thickness and concentration over one timestep:

DIFF1 = 0.004 * DELTXA,

where DELTXA is the x (longitude) grid spacing in meters for thermodynamic fields. This term will be used later in determining the advection of the ice thickness and compactness.

To calculate the growth rates of the ice, the demarcation between thick and thin ice is defined as HO = 0.5 * 2.0.

CSU XSUM is called to calculate the total basin ice thickness except at outflow boundary gridpoints.

The initial value of THETA is set to 1.0. This is used later in the RELAX CSU to indicate a backward timestep.

CSU FORM is called to calculate the basic parameters for the RELAX CSU. The ice drift, ocean currents, marine winds, outflow, thermodynamic land/sea mask, ice thickness, and ice

concentration are used to determine the average ice mass in a grid square, the symmetric and asymmetric water drags, the forcing fields, the ice pressure, and viscosities.

The average ice mass per grid square (AMASS), the nonlinear bulk viscosity (ZETA), and the nonlinear shear viscosity (ETA) are defined for each velocity field:

$$\text{AMASS}(I,J) = 0,$$
$$\text{ZETA}(I,J) = \text{HEFF}(I,J,1) * 10^{11},$$
$$\text{ETA}(I,J) = \text{ZETA}(I,J)/4.0,$$

where HEFF is the mean ice thickness per grid cell.

The ice momentum balance equation is solved by calling the RELAX CSU. The ice advection term is also computed there.

The standard predictor corrector iteration scheme is used to center the nonlinear terms. The process begins by looping over the number of timesteps. First a prediction is done followed by a regular timestep. For the prediction, the third time level value for each x and y component of the ice drift is set to the value at the first time level. The intermediate x and y components of the ice drift are set to the x and y components of the ice drift at the first time level:

$$\text{UICE}(I,J,3) = \text{UICE}(I,J,1)$$
$$\text{VICE}(I,J,3) = \text{VICE}(I,J,1)$$
$$\text{UICEC}(I,J) = \text{UICE}(I,J,1)$$
$$\text{VICEC}(I,J) = \text{VICE}(I,J,1)$$

The indicator for the backward timestep THETA is set to 1.0. The FORM CSU is called to calculate the terms for the momentum equation and then the equation is solved by calling the RELAX CSU. This relaxation solution is required to center the nonlinear terms.

After a prediction, a regular timestep is calculated. The first and second time levels of the x and y components of the ice drift are averaged:

$$\text{UICE}(I,J,1) = 0.5 * (\text{UICE}(I,J,1) + \text{UICE}(I,J,2))$$
$$\text{VICE}(I,J,1) = 0.5 * (\text{VICE}(I,J,1) + \text{VICE}(I,J,2))$$

The FORM CSU is called to calculate the terms for the momentum equation. The third time level value for each x and y component of the ice drift is set to the value at the first time level. The intermediate x and y components of the ice drift are set to the x and y components of the ice drift at the first time level. The first time level for each x and y component of the ice drift is set to the value of the second time level:

$$\text{UICE}(I,J,3) = \text{UICE}(I,J,1)$$
$$\text{VICE}(I,J,3) = \text{VICE}(I,J,1)$$
$$\text{UICEC}(I,J) = \text{UICE}(I,J,1)$$
$$\text{VICEC}(I,J) = \text{VICE}(I,J,1)$$
$$\text{UICE}(I,J,1) = \text{UICE}(I,J,2)$$
$$\text{VICE}(I,J,1) = \text{VICE}(I,J,2)$$

The momentum equation is solved by calling the RELAX CSU. This relaxation solution is required to advance to the next timestep.

The t + 1 time values of UICE and VICE are saved for use in the advection of the ice thickness and compactness.

UICEC(I,J) = UICEC(I,J,1),

VICEC(I,J) = VICEC(I,J,1),

where I and J are loop indices for the total number of T grid boxes zonally and meridionally, respectively.

The squared velocity and squared velocity difference between times t+1 and t are calculated. This velocity information is then written to standard output along with the timestep counter and date-time group.

SQ = SQ + UICE(I,J,1) ** 2 + VICE(I,J,1) ** 2

UERR = UICE(I,J,1) – UICE(I,J,2)

VERR = VICE(I,J,1) – VICE(I,J,2)

SQ1 = SQ1 + (UERR * UERR) + (VERR * VERR)

SMU = max(abs(UERR,SMU))

SMV = max(abs(VERR,SMV))

CSU ADVECT (Sec. 3.3.5.1) is called twice—once to compute the advection terms for the continuity equation of ice thickness and once for the advection of ice compactness. The advection is a divergence of ice-drift velocity times the ice thickness or ice concentration.

The forcing fields are used by the HEAT CSU (Sec. 3.3.7.1) to calculate the terms necessary for determining the heat balance. Next the GROWTH CSU (Sec. 3.3.5.3) is called to calculate the changes in ice thickness and ice concentration. The total ice in the basin, excluding outflow cells, is recalculated by calling CSU XSUM (Sec 3.3.6.1).

Several sums are computed for ensuring conservation and monitoring various contributions to the ice changes. For each thermodynamic field, the mean ice thickness per grid cell (HEFF), the fraction of the grid cell covered by thick ice (AREA1), the total growth rate of thick ice (FHEFF), the change in compactness due to freezing or melting (GAREA), additional ice to be melted for mixed-layer balance (HCORR), and the net rate of total open water growth (HDIFF1) are corrected at the outflow grid cells:

HEFF(I,J,1) = HEFF(I,J,1) * OUT(I,J)

HEFF(I,J,2) = HEFF(I,J,2) * OUT(I,J)

HEFF(I,J,3) = HEFF(I,J,3) * OUT(I,J)

AREA1(I,J,1) = AREA1(I,J,1) * OUT(I,J)

AREA1(I,J,2) = AREA1(I,J,2) * OUT(I,J)

AREA1(I,J,3) = AREA1(I,J,3) * OUT(I,J)

FHEFF(I,J) = FHEFF(I,J) * OUT(I,J)

GAREA(*I*,*J*) = GAREA(*I*,*J*) * OUT(*I*,*J*)
HCORR(*I*,*J*) = HCORR(*I*,*J*) * OUT(*I*,*J*)
HDIFF1(*I*,*J*) = HDIFF1(*I*,*J*) * OUT(*I*,*J*)

Also, the sum of the thickness over all grid squares (THEFF2), the sum of ice compactness over all grid squares (ARSUM), and the sum of growth rate of ice over all grid squares (FHSUM) is 0

GRSUM = GRSUM + HEFF1(*I*,*J*)
THEFF2 = THEFF2 + HEFF(*I*,*J*,1)
ARSUM = ARSUM + AREA1(*I*,*J*,1)
FHSUM = FHSUM + FHEFF(*I*,*J*)

Then the net values for all timesteps are determined. These include the net open water growth (GRSUM1), the net ice growth (FHSUM1), net ice concentration (ARSUM1), the outflow for the current timestep (TOUT1), the total basin ice thickness (THEFF), and the net outflow (TOUT):

GRSUM1 = GRSUM1 + GRSUM
FHSUM1 = FHSUM1 + FHSUM
ARSUM1 = ARSUM1 + ARSUM
TOUT1 = THEFF – THEFF2 – THEFF1
THEFF = THEFF2
TOUT = TOUT + TOUT1

If the current timestep interval is one at which to write to standard output, then the values of the date-time group (IDTG), the current timestep (ICOUNT), the squared velocity (THEFF), the squared velocity difference (THEFF1), the outflow for the current timestep (TOUT1), the net outflow (TOUT), ice growth for the current timestep (FHSUM), the net ice growth (FHSUM1), the sum of the thin ice over all grid squares (GRSUM), the net open water growth (GRSUM1), the sum of ice compactness over all grid squares (ARSUM), and the net ice concentration (ARSUM1) are printed.

If the current timestep is the last timestep, then the process is terminated. This iterative process continues until all timesteps are processed.

3.3.2 CSC Atmospheric Forcing

Atmospheric forcing fields are read in CSU RFORCE, which in turn calls CSU GEOWIND to calculate the geostrophic winds.

3.3.2.1 CSU RFORCE

CSU RFORCE reads the atmospheric forcing fields derived from the 1992 results of NOGAPS and calls CSU GEOWIND to calculate the geostrophic winds. Winds are output for use by the CSC Compute Ocean.

3.3.2.1.1 CSU RFORCE Design Specification/Constraints – Atmospheric forcing data produced by the NOGAPS model are read from a file connected to unit 12. There are no known constraints.

3.3.2.1.2 CSU RFORCE Design –

Input Data Elements:

TA	NOGAPS atmospheric forcing – surface air temperature
PSA	NOGAPS atmospheric forcing – surface air pressure
ESA	NOGAPS atmospheric forcing – surface vapor pressure
FSH1	NOGAPS atmospheric forcing – incoming solar radiation (short wave)
PSB	NOGAPS atmospheric forcing – total heat flux
ESB	NOGAPS atmospheric forcing – sensible heat flux

Output Data Elements:

ES	NOGAPS atmospheric forcing – surface vapor pressure
ES1	NOGAPS atmospheric forcing – sensible heat flux
FSH	NOGAPS atmospheric forcing – incoming solar radiation (short wave)
GAIRX	x component of the geostrophic wind
GAIRY	y component of the geostrophic wind
PS	NOGAPS atmospheric forcing – surface air pressure
PS1	NOGAPS atmospheric forcing – total heat flux
TAIR	NOGAPS atmospheric forcing – surface air temperature

Parameters:
stored in file ice.par

The ice.par parameters are described in Sec. 3.1.1.2.

Data element design information is provided in Sec. 4.0.

Local Data Elements:

ESA(IMTP1,JMTP1)	NOGAPS atmospheric forcing – surface vapor pressure	real
ESB(IMTP1,JMTP1)	NOGAPS atmospheric forcing – sensible heat flux	real
FSH1(IMTP1,JMTP1)	NOGAPS atmospheric forcing – incoming solar radiation	real
GAX(IMTP1,JMTP1)	x component of the wind velocity	real
GAY(IMTP1,JMTP1)	y component of the wind velocity	real
I	index counter	integer
IMT	total number of T grid boxes zonally (360)	integer
IMTP1	total number of T grid boxes zonally + 1 (361)	integer
J	index counter	integer
JMT	total number of T grid boxes meridionally (360)	integer
JMTP1	total number of T grid boxes meridionally + 1 (361)	integer
PSA(IMTP1,JMTP1)	NOGAPS atmospheric forcing – surface pressure	real
PSB(IMTP1,JMTP1)	NOGAPS atmospheric forcing – total heat flux	real
TA(IMTP1,JMTP1)	NOGAPS atmospheric forcing – air temperature	real

Logic Flow:

The RFORCE CSU is called by the ICEMDL CSU to read the forcing fields.

The atmospheric forcing fields are read from a file connected to logical unit 12. These include NOGAPS atmospheric forcing values for the air temperature (TA), the surface pressure (PSA), the surface vapor pressure (ESA), the incoming solar radiation (FSH1), the total heat flux (PSB), and the sensible heat flux (ESB).

The CSU GEOWIND is called to calculate the x and y components of the geostrophic wind from the surface pressure fields of atmospheric forcing.

Before returning to the calling CSU, the atmospheric fields are converted into units needed for the ice portion of the model.

Local Data Files:

The NOGAPS atmospheric forcing data are in a permanent, IEEE binary file called p92<*mmdd*>.dat (unit 12), where <*mmdd*> is the current two-digit month and two-digit day.

3.3.2.2 CSU GEOWIND

The GEOWIND CSU calculates geostrophic winds from surface pressure fields.

3.3.2.2.1 CSU GEOWIND Design Specification/Constraints – There are no known constraints.

3.3.2.2.2 CSU GEOWIND Design

Input Data Elements:

PS	NOGAPS atmospheric forcing – surface air pressure

/CORSP/

FCORSP	array of sine of latitude positions for each gridpoint

/STEPSP/

DELTXU	x (longitude) grid spacing for velocity fields (m)
DELTYU	y (latitude) grid spacing for velocity fields (m)

Output Data Elements:

G1X	x component of geostrophic wind
G1Y	y component of geostrophic wind

Parameters:
stored in file ice.par

The ice.par parameters are described in Sec. 3.1.1.2.

Data element design information is provided in Sec. 4.0.

Local Data Elements:

CMBNM	conversion factor from millibars to N/m^2 (100.0 N/m^2)	real
F	Coriolis parameter (1.46×10^{-4}/s)	real
FPIN	constant used in calculating the x and y components of the geostrophic wind (1.0/(F * RHO))	real
I	index counter	integer
J	index counter	integer
RHO	density of air (1.3 km/m^3)	real

Algorithms:

To calculate the x ($G1X$) and y ($G1Y$) components of the geostrophic wind, the following equations are used:

$$FPIN = 1.0 / (F * RHO) \tag{6}$$

$$G1X_{i+1,j+1} = -1.0 * FPIN * CMBNM * ((PS_{i,j+1} + PS_{i+1,j+1}) / 2 - (PS_{i,j} + PS_{i+1,j}) / 2) / DELTYU / FCORSP_{i,j} \tag{7}$$

$$G1Y_{i+1,j+1} = FPIN * CMBNM * ((PS_{i+1,j} + PS_{i+1,j+1}) / 2 \quad (8)$$
$$- (PS_{i,j} + PS_{i,j+1}) / 2) / DELTXU_j / FCORSP_{i,j},$$

where F is the Coriolis parameter (1.46×10^{-4}/s), *RHO* is the density of air (1.3 km/m^3), *CMBNM* is the conversion factor from millibars to N/m^2 (100.0 N/m^2), *PS* is the pressure field of atmospheric forcing, *DELTYU* is the y (latitude) grid spacing for velocity fields in meters, *DELTXU* is the x (longitude) grid spacing for velocity fields in meters, and *FCORSP* is the latitude position of the grid element. *F*, *RHO*, and *CMBNM* are local constants. *PS*, *DELTYU*, *DELTXU*, and *FCORSP* are inputs to the CSU. The values of *G1X* and *G1Y* are returned to the calling CSU.

Data Conversions:

The pressure must be converted from millibars to N/m^2.

Logic Flow:

The RFORCE CSU calls the GEOWIND CSU to calculate the geostrophic winds from the surface pressure fields. The surface air pressure (PS), the array of sine of latitude positions (FCORSP), the x and y grid spacing (DELTXU and DELTYU) are all inputs to the CSU and are used in determining the geostrophic winds.

Before any calculations, the constant values for the Coriolis parameter (F), the density of air (RHO), and the conversion factor from millibars to N/m^2 (CMBNM) are defined:

F = 1.46 * 10^{-4}

RHO = 1.3

FPIN = 1.0/(F * RHO)

CMBNM = 100.0.

Then the x and y components of the inner geostrophic wind values for each velocity field are defined:

G1X(*I*+1,*J*+1) = −1.0 * FPIN * CMBNB * ((PSI(*I*,*J*+1) + PS(*I*+1,*J*+1)) * 0.5
− (PS(*I*,*J*) + PS(*I*+1,*J*)) * 0.5)/DELTYU/FCORSP(*I*,*J*)

G1X(*I*+1,*J*+1) = FPIN * CMBNB * ((PSI(*I*+1,*J*) + PS(*I*+1,*J*+1)) * 0.5
− (PS(*I*,*J*) + PS(*I*,*J*+1)) * 0.5)/DELTXU(*J*)/FCORSP(*I*,*J*).

The boundary values are set equal to inside values by first looping over the zonal portion of the grid boxes (*I*):

G1X(*I*,1) = G1X(*I*,2)

G1Y(*I*,1) = G1Y(*I*,2)

G1X(*I*,NY2) = G1X(*I*,NY1)

G1Y(I,NY2) = G1Y(I,NY1),

and then looping over the meridian portion of the grid boxes (*J*)

G1X(1,*J*) = G1X(2,*J*)

G1Y(1,*J*) = G1Y(2,*J*)

G1X(NX2,*J*) = G1X(NX1,*J*)
G1Y(NX2,*J*) = G1Y(NX1,*J*).

The x and y components of the geostrophic winds are returned to CSU RFORCE through the parameter list.

3.3.3 CSC Momentum Balance

The Momentum Balance CSC calculates the terms needed to solve the momentum equation. The CSUs listed below perform the functions required by the CSC Momentum Balance.

FORM	Computes the x and y components of forcing due to the ocean currents plus the ice pressure gradient.
RELAX	Uses the relaxation method to solve the momentum balance equation.
QMAX	Finds the maximum value in an array of values.

The forcing due to the ocean current is estimated in the FORM CSU. The x and y components of forcing are then used by the RELAX CSU to solve the momentum equation through the relaxation method. The CSU QMAX is called by the RELAX CSU to find the largest U and V correction values for each of the grids.

3.3.3.1 CSU FORM

The FORM CSU calculates the forcing terms for the ice momentum balance equation. The "law of the wall" is used to estimate the water/ice drag coefficient and the corresponding turning angle between the top mixed layer and the ice bottom.

3.3.3.1.1 CSU FORM Design Specification/Constraints – There are no known constraints.

3.3.3.1.2 CSU FORM Design

Input Data Elements:

AREA1	fraction of grid cell covered by ice
ETA	nonlinear shear viscosity
GAIRX	x component of the geostrophic wind
GAIRY	y component of the geostrophic wind
GWATX	x component of ocean current
GWATY	y component of ocean current
HEFF	mean ice thickness per grid cell
OUT	land/sea mask including outflow conditions for thermodynamic variables
UICE	x component of the ice drift
VICE	y component of the ice drift
ZETA	nonlinear bulk viscosity

/FCORSP/

FCORSP	array of sine of latitude positions for each gridpoint

/STEPSP/

DELTXU	x (longitude) grid spacing for velocity fields (m)
DELTYU	y (latitude) grid spacing for velocity fields (m)

Output Data Elements:

AMASS	ice mass per grid area
DRAGA	asymmetric water drag plus the Coriolis parameter
DRAGS	symmetric water drag
ETA	nonlinear shear viscosity
HEFFM	land/sea mask for thermodynamic variables
ZETA	nonlinear bulk viscosity

/FORCE/

FORCEX	x component of the forcing due to the ocean currents plus the ice pressure gradient
FORCEY	y component of the forcing due to the ocean currents plus the ice pressure gradient

/PRESS/

PRESS	ice strength

Parameters:
stored in file ice.par

The ice.par parameters are described in Sec. 3.1.1.2.

Data element design information is provided in Sec. 4.0.

Local Data Elements:

CDWI	variable drag coefficient between ice and water	real
CONSTK	von Karman constant (0.41)	real
CONSTZ	minimum ice roughness (0.01)	real
COR(IMTM1,JMTM1)	array of the Coriolis forcing	real
COSWAT	cosine of the turning angle for water (0.9063)	real
COSWIN	cosine of the turning angle for wind (0.9205)	real
DAIRN	nonlinear wind drag	real
DWATN(IMTM1,JMTM1)	nonlinear water drag	real
ECCEN	ratio of the principal axes of the plastic yield ellipse (2.0)	real
FCOR	Coriolis parameter (1.46×10^{-4}/s)	real
HMIN	minimum ice thickness	real
I	index counter	integer
II	index counter	integer
J	index counter	integer
JJ	index counter	integer
RHOAIR	density of air (1.3 kg/m^3)	real
SINWAT	sine of the turning angle for water (0.4226)	real
SINWIN	sine of the turning angle for wind (0.3907)	real

Algorithms:

The ice mass per grid cell m is calculated by:

$$m = \rho_i h \,, \tag{9}$$

where ρ_i is the density of ice 0.91×10^3 kg/m^3 and h is the ice thickness as input to the CSU. The calculated ice mass is output by the CSU and is also used by this CSU to calculate the Coriolis term.

After computing the ice mass, the Coriolis term for each grid cell is calculated using:

$$COR = mf \sin(lat), \tag{10}$$

where m is the ice mass calculated above, f is the constant Coriolis parameter value of 1.46×10^{-4}/sec and *lat* is the latitude of the grid element in radians.

The equation for air stress is as follows:

$$\tau_a = \rho_a C_a \left| \vec{U}_g \right| \left(\vec{U}_g \cos\phi + \hat{k} * \vec{U}_g \sin\phi \right), \tag{11}$$

where ρ_a is the density of air, C_a is the air drag coefficient, $\vec{U}_g$ is the geostrophic wind, $\hat{k}$ is a unit vector normal to the surface, and ϕ is the air turning angle.

The equation for water stress is as follows:

$$\tau_w = \rho_w C_w \left| \vec{U}_w - \vec{u} \right| \left[\left(\vec{U}_w - \vec{u} \right) \cos\theta + \hat{k} * \left(\vec{U}_w - \vec{u} \right) \sin\theta \right], \tag{12}$$

where ρ_w is the density of water, C_w is the water drag coefficient, $\vec{U}_w$ is the geostrophic current, $\vec{u}$ is the ice-drift velocity, θ is the water turning angle, and $\hat{k}$ is a unit vector normal to the surface.

The ice strength is calculated by using the following equation:

$$P = P^* \, h \exp[-C(1 - A)], \tag{13}$$

where P^* is the pressure constant (2.75×10^4 N/m^2), C is a fixed empirical constant (20), h is the ice thickness, and A is the ice compactness. This relationship allows the ice to become stronger as it thickens.

Data Conversions:

Latitude positions must be converted from degrees to radians before calculations may be performed.

Logic Flow:

The FORM CSU is called by the ICEMDL CSU. It is dedicated to determining the x and y components of forcing due to the ocean currents plus the ice pressure gradient. These two equations are stated as:

FORCEX = X(wind stress) + X(water drag) + X(tilt) + X(ice pressure gradient)

FORCEY = Y(wind stress) + Y(water drag) + Y(tilt) + Y(ice pressure gradient).

Each x and y component (wind stress, water drag, tilt, and ice pressure gradient) is solved and added to the previous value of *FORCEX* and *FORCEY*, respectively.

The process of finding the ice pressure gradient and water drag begins by calculating the average ice mass per grid cell (AMASS) and the Coriolis term (COR) for each grid cell. The ice mass per grid cell is calculated by:

$$AMASS(I,J) = 0.91 * 10^3 * 0.25 * (HEFF(I,J,1) + HEFF(I+1,J,1) + HEFF(I,J+1,1) + HEFF(I+1,J+1,1)),$$

where HEFF is the mean ice thickness. The Coriolis term for each grid cell is based on the ice mass, Coriolis parameter, and latitude and is computed using:

$$COR(I,J) = AMASS(I,J) * FCOR * FCORSP(I,J),$$

where AMASS is the ice mass per grid cell, FCOR is the Coriolis parameter constant value of 1.46×10^{-4}/s and FCORSP is the latitude of the gridpoint in degrees.

Before calculation of the nonlinear water drag, the minimum ice thickness (HMIN) is calculated, values are assigned for von Karman's constant (CONSTK) and the minimum ice roughness (CONSTZ), and value of the variable drag coefficient between ice and water (CDWI) is calculated as follows:

```
HMIN = max (HEFF(I,J,1), 0.015),
CONSTK = 0.41,
CONSTZ = 0.01,
CDWI = 1.0/(alog(HMIN/CONSTZ)/CONSTK) ** 2,
DWATN(I,J) = 1000 * sqrt((UICE(I,J,1) - GWATX(I,J)) ** 2
                + (VICE(I,J,1) - GWATY(I,J)) ** 2) * CDWI,
```

where UICE is the x component of the ice drift, VICE is the y component of the ice drift, GWATX is the x component of the ocean current, and GWATY is the y component of the ocean current. This calculated value of DWATN is set to 0.055 if <0.055.

Although the asymmetric (DRAGA) and symmetric (DRAGS) water drags are not used in calculating the forcing fields, the values to be passed to subroutine relax are calculated here using

```
DRAGA(I,J) = DWATN(I,J) * SINWAT + COR(I,J),
DRAGS(I,J) = DWATN(I,J) * COSWAT,
```

where SINWAT (0.4226) is the sine of the turning angle for water (25°) and COSWAT (0.9063) is the cosine of the turning angle for water (25°).

Processing is continued to solve for the wind stress. The nonlinear wind DAIRN is calculated by

```
DAIRN = RHOAIR * 0.0008 * sqrt(GAIRX(I+1,J+1) ** 2 + GAIRY(I+1,J+1) ** 2) ,
```

where RHOAIR is the density of air (1.3 kg/m^3), 0.0008 is the wind drag coefficient, GAIRX is the x component of the geostrophic wind, and GAIRY is the y component of the geostrophic wind

Once all values necessary for the x and y components due to wind stress are calculated, the initial values for x and y components of forcing due to ocean currents and pressure gradient are computed as follows:

```
FORCEX(I,J) = DAIRN(I,J) * (COSWIN * GAIRX(I+1,J+1) - SINWIN * GAIRY(I+1,J+1))
FORCEY(I,J) = DAIRN(I,J) * (SINWIN * GAIRX(I+1,J+1) + COSWIN * GAIRY(I+1,J+1)) ,
```

where GAIRX is the x component of the geostrophic wind, GAIRY is the y component of the geostrophic wind, SINWIN (0.3907) is the sine of the turning angle of air (23°), and COSWIN (0.9205) is the cosine of the turning angle of air (23°).

Next, the water stress is added to the existing forcing fields:

$$\begin{aligned} FORCEX(I,J) = FORCEX(I,J) + DWATN(I,J) * (COSWAT * GWATX(I,J) \\ - SINWAT * GWATY(I,J)), \end{aligned}$$

$$\begin{aligned} FORCEY(I,J) = FORCEY(I,J) + DWATN(I,J) * (SINWAT * GWATX(I,J) \\ - COSWAT * GWATY(I,J)) , \end{aligned}$$

where GWATX is the x component of the ocean current, GWATY is the y component of the ocean current, COSWAT (0.9063) is the cosine of the turning angle of water (25°), and SINWAT (0.4226) is the sine of the turning angle of water (25°).

The tilt (or water pressure gradient) in the ice model is computed from the Coriolis parameter and the geostrophic current:

$$FORCEX(I,J) = FORCEX(I,J) - COR(I,J) * GWATY(I,J),$$

$$FORCEY(I,J) = FORCEY(I,J) + COR(I,J) * GWATX(I,J) ,$$

where GWATX is the x component of the ocean current and GWATY is the y component of the ocean current.

The ice pressure (strength) is calculated using:

$$PRESS(I,J) = 2.75 * 10^4 * HEFF(I,J,1) * \exp(-20.0 * (1.0 - AREA1(I,J,1))) ,$$

where HEFF is the mean ice thickness per grid cell and AREA1 is the area of the grid cell covered by ice.

CSU PLAST is called to calculate strain rates (E11, E12, and E22), divergence (DIV), and nonlinear bulk and shear viscosities (ZETA and ETA). After returning from the PLAST CSU, the viscosities and pressure are set equal to 0 at the outflow gridpoints:

$$ETA(I,J) = ETA(I,J) * OUT(I,J),$$

$$ZETA(I,J) = ZETA(I,J) * OUT(I,J),$$

$$PRESS(I,J) = PRESS(I,J) * OUT(I,J) ,$$

where OUT is the land/sea mask including outflow conditions for thermodynamic variables.

The final step to solving the equation for the x and y components of the forcing is to subtract the ice pressure gradient:

$$\begin{aligned} FORCEX(I,J) = FORCEX(I,J) - (1.0/DELTXU(J)/2.0) * PRESS(I+1,J) \\ - PRESS(I,J) + PRESS(I+1,J+1) - PRESS(I,J+1))/2.0 \end{aligned}$$

$$\begin{aligned} FORCEY(I,J) = FORCEY(I,J) - (1.0/DELTYU/2.0) * PRESS(I,J+1) \\ - PRESS(I,J) + PRESS(I+1,J+1) - PRESS(I+1,J))/2.0 , \end{aligned}$$

where DELTXU is the x grid spacing in meters for the velocity fields and DELTYU is the y grid spacing in meters for the velocity fields.

This CSU FORM returns the nonlinear shear and bulk viscosities (ETA and ZETA), the ice mass per grid area (AREA1), the symmetric and asymmetric water drags (DRAGS and DRAGA), and the x and y components of forcing (FORCEX and FORCEY) to the ICE CSU upon completion.

3.3.3.2 CSU RELAX

The RELAX CSU uses the method of relaxation to solve the momentum equation. This is a successive approximation method for solving systems of equations where the errors from an initial approximation are viewed as constraints to be minimized or relaxed within a toleration limit.

3.3.3.2.1 CSU RELAX Design Specification/Constraints – There are no known constraints.

3.3.3.2.2 CSU RELAX Design

Input Data Elements:

AMASS	ice mass per grid cell
DRAGA	asymmetric water drag plus the Coriolis parameter
DRAGS	symmetric water drag
ERROR	maximum error allowed in the relation scheme
ETA	nonlinear shear viscosity
HEFFM	land/sea mask for thermodynamic variables
THETA	indicates a backward timestep
UICE	x component of the ice drift
UICEC	intermediate x component of the ice drift
UVM	land/sea mask for the velocity variables
VICE	y component of the ice drift
VICEC	intermediate y component of the ice drift
ZETA	nonlinear bulk viscosity

/CORSP/

FCORSP	array of sine of latitude positions for each gridpoint

/FORCE/

FORCEX	x component of forcing due to the ocean currents plus the ice pressure gradien
FORCEY	y component of forcing due to the ocean currents plus the ice pressure gradien

/STEP/

DELTAT	timestep (s)
DELTAY	y grid spacing (deg)

/STEPSP/

DELTXU	x grid spacing for velocity fields (m)
DELTYU	y grid spacing for velocity fields (m)

Output Data Elements:

UICE	x component of the ice drift
VICE	y component of the ice drift

Parameters:
stored in file ice.par

The ice.par parameters are described in Sec. 3.1.1.2.

stored in file relax.par

N3	360 * 360
N4	359 * 359
MX	359
MY	359
M4X	360/2
M4Y	360/2
M4XY	M4X * M4Y
M4XP	M4X+1
M4XM	M4X–1
M1STOP	(359/2–1)M4X + 359/2
M2STOP	(359/2–1)M4X + 360/2–1
M3STOP	(360/2–2)M4X + 359/2
MSTOP	(360/2–1)M4X – 1

Data element design information is provided in Sec. 4.0.

Local Data Elements:

COEFIX(IMTM1,JMTM1)	used in determining relaxation coefficients 19, 20, and 21	real
COEFIY(IMTM1,JMTM1)	used in determining relaxation coefficients 18, 19, and 21	real
DELIN	reciprocal of y grid spacing (m)	real
DELIN2	used to define S1TH in spherical coordinates	real
DRAG1	used in calculating relaxation coefficients 18, 19, and 21	real
DRAG2	used in calculating relaxation coefficients 19, 20, and 21	real
FX3(M4XY)	array for 4-color solver	real
FX3A	added to FX3 for spherical coordinate system	real
FXM(M4XY,4)	array for 4-color solver	real
FY3(M4XY)	array for 4-color solver	real
FY3A	added to FY3 for spherical coordinate system	real
FYM(M4XY,4)	array for 4-color solver	real
I	index counter	integer
I4	temporary variable	integer
ICOUNT	counter for the number of iterations	integer
IJ4	index of first dimension for relaxation coefficients	integer
J	row index counter	integer
J4	temporary variable	integer
K	index of loop over four grids	integer
K4	index of third dimension for relaxation coefficients	integer
R(M4XY,21,4)	relaxation coefficients	real
RADIAN	degrees per radian (57.29578°)	real
RADIUS	radius of the earth (6370.0×10^3 m)	real
S1	maximum error for u and v	real
S11	largest value of the u velocity for grid 1	real
S12	largest value of the u velocity for grid 2	real
S13	largest value of the u velocity for grid 3	real

S14	largest value of the u velocity for grid 4	real
S1TH	used in defining relaxation coefficients 5, 6, 7, 8, 9, 10, 11, 12, and 13	real
S21	largest value of the v velocity for grid 1	real
S22	largest value of the v velocity for grid 2	real
S23	largest value of the v velocity for grid 3	real
S24	largest value of the v velocity for grid 4	real
S2TH	used in defining relaxation coefficients 1, 4, 14, and 17	real
U4(M4XY,4)	x component of the ice drift	real
U4C(M4XY)	u velocity correction	real
V4(M4XY,4)	y component of the ice drift	real
V4C(M4XY)	v velocity correction	real
WFA	relaxation factor (1.5)	real

Algorithms:

Given the present ice rheology, there are 17 independent coefficients necessary for the relaxation steps. These coefficients are calculated by using the following equations:

$$R_1 = (1/2/dy/dy)(\eta_{i,j} + \eta_{i+1,j}) + (m_{i,j}\ vc_{i,j}/dy/2)$$

$$R_2 = (1/2/dx_j/dx_j)(\eta_{i,j+1} + \eta_{i,j} + \zeta_{i,j+1} + \zeta_{i,j}) + m_{i,j}\ uc_{i,j}/dx_j/2$$

$$R_3 = (1/2/dx_j/dx_j)(\eta_{i+1,\,j+1} + \eta_{i+1,j} + \zeta_{i+1,\,j+1} + \zeta_{i+1,j}) - m_{i,j}\ uc_{i,j}/dx_j/2$$

$$R_4 = (1/2/dy/dy)(\eta_{i+1,\,j+1} + \eta_{i,j+1}) - (m_{i,j}\ vc_{i,j}/dy/2)$$

$$R_5 = (1/4/dx_j/dy)(\zeta_{i,j})$$

$$R_6 = (1/4/dx_j/dy)[2(-\eta_{i,j} - \eta_{i+1,j}) - (-\zeta_{i,j} + \zeta_{i+1,j})]$$

$$R_7 = (1/4/dx_j/dy)(-\zeta_{i+1,j})$$

$$R_8 = (1/4/dx_j/dy)[2(-\eta_{i,j+1} - \eta_{i,j}) - (-\zeta_{i,j+1} + \zeta_{i,j})]$$

$$R_9 = (1/4/dx_j/dy)(\zeta_{i+1,j} + \zeta_{i,j} - \zeta_{i,j} - \zeta_{i+1,\,j+1})$$

$$R_{10} = (1/4/dx_j/dy)[2(\eta_{i+1,\,j+1} - \eta_{i+1,j}) - (\zeta_{i+1,\,j+1} - \zeta_{i+1,j})]$$

$$R_{11} = (1/4/dx_j/dy)(-\zeta_{i,j+1})$$

$$R_{12} = (1/4/dx_j/dy)[2(\eta_{i,j+1} - \eta_{i+1,\,j+1}) - (\zeta_{i,j+1} - \zeta_{i+1,\,j+1})]$$

$$R_{13} = (1/4/dx_j/dy)(\zeta_{i+1,\,j+1})$$

$$R_{14} = (1/2/dy/dy)(\eta_{i,j} + \eta_{i+1,j} + \zeta_{i,j} + \zeta_{i+1,j}) + m_{i,j}\ vc_{i,j}/dy/2$$

$$R_{15} = (1/2/dx_j/dx_j)(\eta_{i,j} + \eta_{i,j+1}) + (m_{i,j}\ uc_{i,j}/dx_j/2)$$

$$R_{16} = (1/2/dx_j/dx_j)(\eta_{i+1,\,j+1} + \eta_{i+1,j}) - m_{i,j}\ uc_{i,j}/dx_j/2$$

$$R_{17} = (1/2/dy/dy)(\eta_{i+1,\,j+1} + \eta_{i,j+1} + \zeta_{i+1,\,j+1} + \zeta_{i,j+1}) - m_{i,j}\ vc_{i,j}/dy/2\,. \qquad (14)$$

Here, *dy* is the y grid spacing in meters, *dx* is the x grid spacing in meters, η is the nonlinear shear viscosity, ζ is the nonlinear bulk viscosity, *vc* is the intermediate y component of the ice drift for use in the semi-implicit timestepping, *uc* is the intermediate x component of the ice drift for use in the semi-implicit timestepping, and *m* is the ice mass per grid area.

To perform the relaxation, the 17 independent coefficients are used to calculate the following equations:

$$RHS_x = FXM_{i,j} + \begin{vmatrix} 0 & R_4 & 0 \\ R_2 & 0 & R_3 \\ 0 & R_1 & 0 \end{vmatrix} U_{i,j} + \begin{vmatrix} R_{11} & R_{12} & R_{13} \\ R_8 & R_9 & R_{10} \\ R_5 & R_6 & R_7 \end{vmatrix} V_{i,j}$$

$$RHS_y = FYM_{i,j} + \begin{vmatrix} R_{11} & -R_{12} & R_{13} \\ -R_8 & R_9 & -R_{10} \\ R_5 & -R_6 & R_7 \end{vmatrix} U_{i,j} + \begin{vmatrix} 0 & R_{17} & 0 \\ R_{15} & 0 & R_{16} \\ 0 & R_{14} & 0 \end{vmatrix} V_{i,j} \quad . \qquad (15)$$

Data Conversions:

Latitude positions are converted from degrees to radians.

Logic Flow:

The RELAX CSU is called by the ICEMDL CSU.

Upon entering this CSU, the values of R, FXM, FYM, U4, V4, U4C, V4C, FX3, and FY3 are initialized to 0. The value of ICOUNT is set to 0 and the relaxation factor WFA is set to 1.5. DELIN is set to the reciprocal of the y (latitude) grid spacing:

```
DELIN = 1.0/DELTYU
DELIN2 = DELIN * DELIN/2.0
```

Then the x and y components of the ice drift at the second time level are set equal to the one at the first:

```
UICE(I,J,2) = UICE(I,J,1)
VICE(I,J,2) = VICE(I,J,1)
```

The ice drift at the boundary points are set equal to 0 by multiplying the ice drift by the land/sea mask:

```
UICE(I,J,1) = UICE(I,J,3) * UVM(I,J)
VICE(I,J,1) = VICE(I,J,3) * UVM(I,J)
```

COEFFIX and COEFFIY are defined in spherical coordinates as

```
COEFIX(I,J) = 1.0/AMASS(I,J)/DELTAT
            + THETA * (DRAGS(I,J) + ETA(I,J) + ETA(I+1,J)
```

$$+ \text{ETA}(I,J+1) + \text{ETA}(I+1,J+1))/2.0 \ \text{DELTYU} ** 2$$
$$+ (\text{ETA}(I,J) + \text{ETA}(I+1,J) + \text{ETA}(I,J+1) + \text{ETA}(I+1,J+1) + \text{ZETA}(I,J)$$
$$+ \text{ZETA}(I+1,J) + \text{ZETA}(I,J+1) + \text{ZETA}(I+1,J+1))/2.0/\text{DELTXU}(J) ** 2))$$

$$\text{COEFIY}(I,J) = 1.0/(\text{AMASS}(I,J)/\text{DELTAT}$$
$$+ \text{THETA} * (\text{DRAGS}(I,J) + \text{ETA}(I,J) + \text{ETA}(I+1,J) + \text{ETA}(I,J+1)$$
$$+ \text{ETA}(I+1,J+1))/2.0/\text{DELTXU}(J) ** 2$$
$$+ (\text{ETA}(I,J) + \text{ETA}(I+1,J) + \text{ETA}(I,J+1)$$
$$+ \text{ETA}(I+1,J+1) + \text{ZETA}(I,J) + \text{ZETA}(I+1,J) + \text{ZETA}(I,J+1)$$
$$+ \text{ZETA}(I+1,J+1))/2.0/\text{DELTYU} ** 2)),$$

where THETA = 1 for the Euler backward iteration.

The values of FY3A and FX3A are 0

$$\text{FY3A} = -\tan(\text{DELTAY} * (J-(\text{MIDY}-0.5) - 0.5)/\text{RADIAN})/\text{RADIUS}) * 2.0$$
$$* ((\text{ETA}(I,J) + \text{ETA}(I-1,J) + \text{ETA}(I,J-1) + \text{ETA}(I-1,J-1))/4.0)$$
$$* ((\text{VICE}(I,J+1,1) - \text{VICE}(I,J-1,1))/2.0/\text{DELTYU}) * \text{UVM}(I,J)$$

$$\text{FX3A} = -\tan(\text{DELTAY} * (J-(\text{MIDY}-0.5) - 0.5)/\text{RADIAN})/\text{RADIUS})$$
$$* ((\text{ETA}(I,J) + \text{ETA}(I-1,J) + \text{ETA}(I,J-1) + \text{ETA}(I-1,J-1))/4.0)$$
$$* ((\text{VICE}(I+1,J,1) - \text{VICE}(I-1,J,1))/2.0/\text{DELTXU}(J)$$
$$+ (\text{UICE}(I,J+1,1) \ \text{UICE}(I,J-1,1))/2.0/\text{DELTYU}) * \text{UVM}(I,J)$$

$$\text{FXM}(IJ4,K4) = \text{UICE}(I,J,2) * \text{AMASS}(I,J)/\text{DELTAT} + \text{FORCEX}(I,J) + \text{FX3A}$$

$$\text{FYM}(IJ4,K4) = \text{VICE}(I,J,2) * \text{AMASS}(I,J)/\text{DELTAT} + \text{FORCEY}(I,J) + \text{FY3A},$$

where DELTAY is the y grid spacing in degrees, MIDY is equal to 160.0, RADIAN is the conversion factor from degrees to radians, RADIUS is the radius of the earth in meters, ETA is the nonlinear shear viscosity, VICE is the y component of the ice drift, DELTYU is the y grid spacing in meters, UVM is the land/sea mask for the velocity fields, DELTXU is the x grid spacing in meters, and UICE is the x component of the ice drift.

At this point, the relaxation coefficients are explicitly calculated. S1TH, S2TH, and SDL are then defined in spherical coordinates:

$$\text{S1TH} = 0.5 * \text{DELIN2} * \text{THETA}$$

$$\text{S2TH} = 2.0 * \text{S1TH}$$

$$\text{SDL} = 0.5 * \text{THETA} * \text{DELIN}$$

The relaxation coefficients are defined for four grids ($K4$):

$$J4 = IJ4/\text{M4X} + 1$$

$$I4 = IJ4 - (J4 - 1) * \text{M4X}$$

$$J = 2 * J4 - (4 - K4)/2.$$

If the value of J is greater than 1 or less than the number of rows for the velocity fields, then the value of I is calculated as:

$I = 2 * I4 - \text{mod}(K4,2)$.

If the value of I is greater than 1 or less than the number of columns for the velocity fields, then the relaxation coefficients are computed by:

R(*IJ*4,1,*K*4) = S2TH * (ETA(*I*,*J*) + ETA(*I*+1,*J*)) + SDL * AMASS(*I*,*J*) * VICEC(*I*,*J*)

R(*IJ*4,2,*K*4) = (ETA(*I*,*J*+1) + ETA(*I*,*J*) + ZETA(*I*,*J*+1) + ZETA(*I*,*J*))
* THETA/(2.0 * DELTXU(*J*) * DELTXU(*J*)) + AMASS(*I*,*J*)
* UICEC(*I*,*J*) * THETA/(2.0 * DELTXU(*J*))

R(*IJ*4,3,*K*4) = (ETA(*I*+1,*J*+1) + ETA(*I*+1,*J*) + ZETA(*I*+1,*J*+1) + ZETA(*I*+1,*J*))
* THETA/(2.0 * DELTXU(*J*) * DELTXU(*J*)) – AMASS(*I*,*J*)
* UICEC(*I*,*J*) * THETA/(2.0 * DELTXU(*J*))

R(*IJ*4,4,*K*4) = S2TH * (ETA(*I*+1,*J*+1) + ETA(*I*,*J*+1)) – AMASS(*I*,*J*) * VICEC(*I*,*J*) * SDL.

The S1TH term is redefined in spherical coordinates as follows:

S1TH = THETA * DELIN/4.0/DELTXU(*J*)

R(*IJ*4,5,*K*4) = S1TH * ZETA(*I*,*J*)

R(*IJ*4,6,*K*4) = – S1TH * (2.0 * (ETA(*I*,*J*) – ETA(*I*+1,*J*))+ZETA(*I*+1,*J*) – ZETA(*I*,*J*))

R(*IJ*4,7,*K*4) = – S1TH * ZETA(*I*+1,*J*)

R(*IJ*4,8,*K*4) = – S1TH * (2.0 * (ETA(*I*,*J*+1) – ETA(*I*,*J*))+ZETA(*I*,*J*) – ZETA(*I*,*J*+1))

R(*IJ*4,9,*K*4) = S1TH * (ZETA(*I*+1,*J*)+ZETA(*I*,*J*+1) – ZETA(*I*,*J*) – ZETA(*I*+1,*J*+1))

R(*IJ*4,10,*K*4) = S1TH * (2.0 * (ETA(*I*+1,*J*+1) – ETA(*I*+1,*J*)) + ZETA(*I*+1,*J*)
– ZETA(*I*+1,*J*+1)))

R(*IJ*4,11,*K*4) = – S1TH * ZETA(*I*,*J*+1)

R(*IJ*4,12,*K*4) = S1TH * (2.0 * (ETA(*I*,*J*+1) – ETA(*I*+1,*J*+1)) + (ZETA(*I*+1,*J*+1)
– ZETA(*I*,*J*+1)))

R(*IJ*4,13,*K*4) = S1TH * ZETA(*I*+1,*J*+1)

R(*IJ*4,14,*K*4) = S2TH * (ETA(*I*,*J*) + ETA(*I*+1,*J*) + ZETA(*I*,*J*) + ZETA(*I*+1,*J*))
+ SDL * AMASS(*I*,*J*) * VICEC(*I*,*J*)

R(*IJ*4,15,*K*4) = (ETA(*I*,*J*) + ETA(*I*,*J*+1) * THETA/(2.0 * DELTXU(*J*)
* DELTXU(*J*)) + AMASS(*I*,*J*) * UICEC(*I*,*J*) * THETA/(2.0
* DELTXU(*J*))

R(*IJ*4,16,*K*4) = (ETA(*I*+1,*J*+1) + ETA(*I*+1,*J*) * THETA/(2.0 * DELTXU(*J*)
* DELTXU(*J*)) – AMASS(*I*,*J*) * UICEC(*I*,*J*) * THETA/(2.0
* DELTXU(*J*))

R(*IJ*4,17,*K*4) = S2TH * (ETA(*I*+1,*J*+1) + ETA(*I*,*J*+1) + ZETA(*I*+1,*J*+1)
+ ZETA(*I*,*J*+1)) – AMASS(*I*,*J*) * VICEC(*I*,*J*) * SDL

DRAG1 = (DRAGA(*I*,*J*) + R(*IJ*4,9,*K*4) * THETA

DRAG2 = (DRAGA(*I*,*J*) – R(*IJ*4,9,*K*4) * THETA

R(*IJ*4,18,*K*4) = THETA * COEFIY(*I*,*J*) * DRAG1

R(*IJ*4,19,*K*4) = COEFIX(*I*,*J*) * UVM(*I*,*J*)/
(1.0+DRAG1 * DRAG2 * COEFIX(*I*,*J*) * COEFIY(*I*,*J*))

R(*IJ*4,20,*K*4) = THETA * COEFIX(*I*,*J*) * DRAG2

R(*IJ*4,21,*K*4) = COEFIY(*I*,*J*) * UVM(*I*,*J*)/
(1.0+DRAG1 * DRAG2 * COEFIX(*I*,*J*) * COEFIY(*I*,*J*)).

Once all of the relaxation coefficients are calculated, the relaxation can be done directly. First, UICE and VICE are loaded into U4 and V4, respectively:

U4(*IJ*4,*K*4) = UICE(*I*,*J*,1)

V4(*IJ*4,*K*4) = VICE(*I*,*J*,1)

The iterative process begins here by performing a relaxation on grid 1:

FX3(*I*) = FXM(*I*,1) + U4(*I*–1,2) * R(*I*,2,1) + U4(*I*,2) * R(*I*,3,1) +
+ U4(*I*–M4X,3) * R(*I*,1,1) + U4(*I*,3) * R(*I*,4,1) + V4(*I*–1,2) * R(*I*,8,1)
+ V4(*I*,2) * R(*I*,10,1) + V4(*I*–M4X,3) * R(*I*,6,1) + V4(*I*,3) * R(*I*,12,1)
+ V4(*I*–M4XP,4) * R(*I*,5,1) + V4(*I*,4) * R(*I*,13,1) + V4(*I*–1,4) * R(*I*,11,1)
+ V4(*I*–M4X,4) * R(*I*,7,1),

FY3(*I*) = FYM(*I*,1) + V4(*I*–1,2) * R(*I*,15,1) + V4(*I*,2) * R(*I*,16,1)
+ V4(*I*–M4X,3) * R(*I*,14,1) + V4(*I*,3) * R(*I*,17,1) – U4(*I*–1,2) * R(*I*,8,1)
– U4(*I*,2) * R(*I*,10,1) – U4(*I*–M4X,3) * R(*I*,6,1) – U4(*I*,3) * R(*I*,12,1)
+ U4(*I*–M4XP,4) * R(*I*,5,1) + U4(*I*,4) * R(*I*,13,1) + U4(*I*–1,4) * R(*I*,11,1)
+ U4(*I*–M4X,4) * R(*I*,7,1),

where *I* = M4XP to M1STOP.

The arrays for the velocity corrections are initialized to 0:

U4C(*I*) = 0,

V4C(*I*) = 0,

where *I* = 1 to M4X.

U4C(*I*) = 0,

V4C(*I*) = 0,

where *I* = M1STOP to M4XY.

The velocity corrections are then calculated:

U4C(*I*) = ((FX3(*I*)+FY3(*I*) * R(*I*,18,1)) * R(*I*,19,1) – U4(*I*,1)) * WFA

V4C(*I*) = ((FY3(*I*) – FX3(*I*) * R(*I*,20,1)) * R(*I*,21,1) – V4(*I*,1)) * WFA,

where *I* = M4XP to M1STOP.

The velocity corrections are then added to the U4 and V4 values for each I = M4XP to M1STOP:

U4(I,1) = U4(I,1) + U4C(I)

V4(I,1) = V4(I,1) + V4C(I)

The CSU QMAX is called to the find the largest correction for both U4C and V4C. The values are returned as S11 and S21, where S11 is the largest correction of the U4C array and S21 is the largest correction of the S21 array.

Once the relaxation on grid 1 is complete, the relaxation is performed on grid 2:

FX3(I) = FXM(I,2) + U4(I,1) * R(I,2,2) + U4(I+1,1) * R(I,3,2)
+ U4(I–M4X,4) * R(I,1,2) + U4(I,4) * R(I,4,2) + V4(I,1) * R(I,8,2)
+ V4(I+1,1) * R(I,10,2) + V4(I–M4X,4) * R(I,6,2) + V4(I,4) * R(I,12,2)
+ V4(I–M4XP,3) * R(I,5,2) + V4(I+1,3) * R(I,13,2) + V4(I,3) * R(I,11,2)
+ V4(I–M4XM,3) * R(I,7,2)

FY3(I) = FYM(I,2) + V4(I,1) * R(I,15,2) + V4(I+1,1) * R(I,16,2)
+ V4(I–M4X,4) * R(I,14,2) + V4(I,4) * R(I,17,2) – U4(I,1) * R(I,8,2)
– U4(I+1,1) * R(I,10,2) – U4(I–M4X,4) * R(I,6,2) – U4(I,4) * R(I,12,2)
+ U4(I–M4X,3) * R(I,5,2) + U4(I+1,3) * R(I,13,2) + U4(I,3) * R(I,11,2)
+ U4(I–M4XM,3) * R(I,7,2),

where I = M4X to M2STOP.

The U4C and V4C array are cleaned up by:

U4C(M1STOP) = 0,

V4C(M1STOP) = 0.

The velocity corrections for grid 2 are then calculated:

U4C(I) = ((FX3(I)+FY3(I) * R(I,18,2)) * R(I,19,2) – U4(I,2)) * WFA,

V4C(I) = ((FY3(I) – FX3(I) * R(I,20,2)) * R(I,21,2) – V4(I,2)) * WFA,

where I = M4X to M2STOP.

The velocity corrections for grid 2 are then added to the U4 and V4 values for each I = M4X to M2STOP:

U4(I,2) = U4(I,2) + U4C(I)

V4(I,2) = V4(I,2) + V4C(I)

The CSU QMAX is called to the find the largest correction for both U4C and V4C. The values are returned as S12 and S22.

Once the relaxation on grid 2 is complete, the relaxation is performed on grid 3:

FX3(I) = FXM(I,3) + U4(I–1,4) * R(I,2,3) + U4(I,4) * R(I,3,3)
+ U4(I,1) * R(I,1,3) + U4(I+M4X,1) * R(I,4,3) + V4(I–1,4) * R(I,8,3)
+ V4(I,4) * R(I,10,3) + V4(I,1) * R(I,6,3) + V4(I+M4X,1) * R(I,12,3)
+ V4(I–1,2) * R(I,5,3) + V4(I+M4X,2) * R(I,13,3)
+ V4(I+M4XM,2) * R(I,11,3) + V4(I,2) * R(I,7,3)

$$FY3(I) = FYM(I,3) + V4(I-1,4) * R(I,15,3) + V4(I,4) * R(I,16,3) + V4(I,1) * R(I,14,3) + V4(I+M4X,1) * R(I,17,3) - U4(I-1,4) * R(I,8,3) - U4(I,4) * R(I,10,3) - U4(I,1) * R(I,6,3) - U4(I+M4X,1) * R(I,12,3) + U4(I-1,2) * R(I,5,3) + U4(I+M4X,2) * R(I,13,3) + U4(I+M4XM,2) * R(I,11,3) + U4(I,2) * R(I,7,3),$$

where I = M4X to M3STOP.

The U4C and V4C array are cleaned up by:

$$U4C(I) = 0,$$

$$V4C(I) = 0,$$

where I = M3STOP to M2STOP.

The velocity corrections for grid 3 are then calculated:

$$U4C(I) = ((FX3(I) + FY3(I) * R(I,18,3)) * R(I,19,3) - U4(I,3)) * WFA,$$

$$V4C(I) = ((FY3(I) - FX3(I) * R(I,20,3)) * R(I,21,3) - V4(I,3)) * WFA,$$

where I = 2 to M3STOP.

The velocity corrections for grid 3 are then added to the U4 and V4 values for each I = 2 to M3STOP:

$$U4(I,3) = U4(I,3) + U4C(I)$$

$$V4(I,3) = V4(I,3) + V4C(I)$$

The CSU QMAX is called to the find the largest correction for both U4C and V4C. The values are returned as S13 and S23.

When the relaxation on grid 3 is complete, the relaxation is performed on grid 4:

$$FX3(I) = FXM(I,4) + U4(I,3) * R(I,2,4) + U4(I+1,3) * R(I,3,4) + U4(I,2) * R(I,1,4) + U4(I+M4X,2) * R(I,4,4) + V4(I,3) * R(I,8,4) + V4(I+1,3) * R(I,10,4) + V4(I,2) * R(I,6,4) + V4(I+M4X,2) * R(I,12,4) + V4(I,1) * R(I,5,4) + V4(I+M4XP,1) * R(I,13,4) + V4(I+M4X,1) * R(I,11,4) + V4(I+1,1) * R(I,7,4)$$

$$FY3(I) = FYM(I,4) + V4(I,3) * R(I,15,4) + V4(I+1,3) * R(I,16,4) + V4(I,2) * R(I,14,4) + V4(I+M4X,2) * R(I,17,4) - U4(I,3) * R(I,8,4) - U4(I+1,3) * R(I,10,4) - U4(I,2) * R(I,6,4) - U4(I+M4X,2) * R(I,12,4) + U4(I,1) * R(I,5,4) + U4(I+M4XP,1) * R(I,13,4) + U4(I+M4X,1) * R(I,11,4) + U4(I+1,) * R(I,7,4),$$

where I = 1 to M4STOP.

The U4C and V4C array are cleaned up by:

$$U4C(M3STOP) = 0,$$

$$V4C(M3STOP) = 0.$$

The velocity corrections for grid 4 are then calculated:

U4C(I) = ((FX3(I)+FY3(I) * R(I,18,4)) * R(I,19,4) – U4(I,4)) * WFA,

V4C(I) = ((FY3(I) – FX3(I) * R(I,20,4)) * R(I,21,4) – V4(I,4)) * WFA,

where I = 1 to M4STOP.

The velocity corrections for grid 4 are then added to the U4 and V4 values for each I = 1 to M4STOP:

U4(I,3) = U4(I,3) + U4C(I)

V4(I,3) = V4(I,3) + V4C(I)

The CSU QMAX is called to the find the largest correction for both U4C and V4C. The values are returned as S14 and S24.

The iteration loop counter ICOUNT is incremented. If the counter is ≤ 2000, then the maximum velocity correction from all four grids is determined. If this value is less than or equal to the maximum error allowed in the relation scheme (ERROR), then the values of UICE and VICE are reloaded; otherwise, the iterative process continues by relaxing on all four grids again. If the number of iterations exceeds 2000, then the convergence fails and a message is sent to standard output. The values of UICE and VICE are reloaded as

```
DO K4 = 1, 4
    DO IJ4 = 1, M4XY
        J4 = IJ4/M4X + 1
        I4 = IJ4 – (J4 – 1) * M4X
        J = 2 * J4 – (4–K4)/2
        IF (.NOT. (J .LE. 1 .OR. J .GE. JMTM1)) THEN
            I = 2 * I4 – MOD(K4,2)
            IF (.NOT. (I .LE. 1 .OR. I .GE. IMTM1) THEN
                UICE(I,J,1) = U4(IJ4,K4)
                VICE(I,J,1) = V4(IJ4,K4)
            END IF
        END IF
    CONTINUE
CONTINUE.
```

The number of iterations and the maximum error for U and V are written to standard output.

3.3.3.3 CSU QMAX

The CSU QMAX finds the maximum value of an array.

3.3.3.3.1 CSU QMAX Design Specification/Constraints – There are no known constraints.

3.3.3.3.2 CSU QMAX Design

Input Data Elements:

U	Input array #1
V	Input array #2

Output Data Elements:

S11	Maximum value in array #1
S21	Maximum value in array #2
U	Input array #1
V	Input array #2

Parameters:

stored in file ice.par

The ice.par parameters are described in Sec. 3.1.1.2.

stored in file qmax.par

M4X	IMT/2; = 180
M4Y	JMT/2; = 180
M4XY	M4X * M4Y; = 32,400

Data element design information is provided in Sec. 4.0.

Local Data Elements:

N	index counter	integer

Logic Flow:

This CSU is called by the RELAX CSU.

The initial maximum values are set to the first element of each array:

U(N) = abs(U(N))

V(N) = abs(V(N)).

The initial maximum values are set to the first element of each array:

S11 = U(1)

S21 = V(1).

The maximum value of each array is then found and control is returned the RELAX CSU:

S11 = max(S11,U(N))

S21 = max(S21,V(N)).

3.3.4 CSC Ice Rheology

The Ice Rheology CSC calculates the ice stress that is directly related to the ice strength and strain rates.

3.3.4.1 CSU PLAST Computer Software Unit

The purpose of the PLAST CSU is to calculate strain rates, divergence, and viscosities based on plastic flow specified by an elliptic yield curve.

3.3.4.1.1 CSU PLAST Design Specification/Constraints – There are no known constraints.

3.3.4.1.2 CSU PLAST Design

Input Data Elements:

ECCEN	ratio of the principal axes of the plastic yield ellipse
ETA	nonlinear shear viscosity
HEFFM	thermodynamic land/sea mask
PRESS	ice strength
UICE	x component of the ice drift
VICE	y component of the ice drift

/STEP/

DELTAX	x (longitude) grid spacing (deg)
DELTAY	y (latitude) grid spacing (deg)

/STEPSP/

DELTXA	x (longitude) grid spacing for thermodynamic fields (m)
DELTYA	y (latitude) grid spacing for thermodynamic fields (m)

Output Data Elements:

ZETA	nonlinear bulk viscosity

Parameters:
stored in file ice.par

The ice.par parameters are described in Sec. 3.1.1.2.

Data element design information is provided in Sec. 4.0.

Local Data Elements:

DELT	used in calculating ETA and ZETA	real
DELT1	square root of DELT	real
E11	xx strain component	real
E12	xy strain component	real
E22	yy strain component	real
ECM2	reciprocal of ECCEN^2	real
GMIN	minimum value of DELT1 (1.0×10^{-20})	real
I	index counter	integer
J	index counter	integer
ZMAX	maximum allowable bulk viscosity value	real
ZMIN	minimum allowable bulk viscosity value (4.0×10^8)	real

Algorithms:

Strain rates, averaged over the grid cell, are estimated for use in solving the momentum equations. The xx $\left(\dot{\varepsilon}_{11}\right)$, yy $\left(\dot{\varepsilon}_{22}\right)$, and xy $\left(\dot{\varepsilon}_{12}\right)$ strain components are obtained by using the following equations:

$$\left(\dot{\varepsilon}_{11}\right)_{i+1/2,\, j+1/2} = \frac{1}{2h}\left[u_{i+1,j+1} + u_{i+1,j} - u_{i,\, j+1} - u_{i,j}\right],$$

$$\left(\dot{\varepsilon}_{22}\right)_{i+1/2,\, j+1/2} = \frac{1}{2h}\left[v_{i+1,j+1} + v_{i,\, j+1} + v_{i+1,j} + v_{i,j}\right],$$

$$\left(\dot{\varepsilon}_{12}\right)_{i+1/2,\, j+1/2} = \frac{1}{4h}\left[u_{i+1,j+1} + u_{i,j+1} - u_{i+1,j} - u_{i,j}\right] + \frac{1}{4h}\left[v_{i+1,j+1} + v_{i+1,j} - v_{i,\, j+1} - v_{i,j}\right], \quad (16)$$

where h is either the x or y grid spacing, u is the x component of the ice drift, and v is the y component of the ice drift.

The nonlinear bulk (ζ) and shear (η) viscosities are calculated using the strain rates from above, the pressure term ($P/2$), and the ratio of the principal axes of the ellipse (e). These terms are stated as

$$\Delta = \left| \left(\dot{\varepsilon}_{11}^2 + \dot{\varepsilon}_{22}^2 \right) \left(1 + 1/e^2 \right) + 4e^{-2} \dot{\varepsilon}_{12}^2 + 2\dot{\varepsilon}_{11}^2 \dot{\varepsilon}_{22}^2 \left(1 - 1/e^2 \right) \right|^{1/2},$$

$$\zeta = P/2\Delta,$$

$$\eta = \zeta/e^2. \tag{17}$$

Logic Flow:

The PLAST CSU is called by the FORM CSU to solve the equations for the strain rates, viscosities, and divergence.

First, the values necessary for solving the viscosity equations are calculated by:

ECM2 = 1.0/(ECCEN^2),

where ECCEN is the ratio of the principal axes of the plastic yield ellipse. Then the constant values for the minimum allowable bulk viscosity (ZMIN) and the minimum allowable value of DELT1 are set:

ZMIN = 4.0 * 10^8

GMIN = 1.0 * 10^{-20}

Next, the xx (E11), yy (E22), and xy (E12) strain rates for each x (I) and y (J) component of the velocity fields are evaluated by averaging the x and y components of the ice drift over the grid cell. The strain rates are used in calculating the nonlinear viscosities:

E11(I,J) = (0.5/DELTAX)
* (UICE(I,J,1) + UICE(I,J–1,1) – UICE(I–1,J,1) – UICE(I–1,J–1,1)),

E22(I,J) = (0.5/DELTAY)
* (VICE(I,J,1) + VICE(I–1,J,1) – VICE(I,J–1,1) – VICE(I–1,J–1,1)),

E12(I,J) = (0.25/DELTAY)
* (UICE(I,J,1) + UICE(I–1,J,1) – UICE(I,J–1,1) – UICE(I–1,J–1,1)),
+ (0.25/DELTAX)
* (VICE(I,J,1) + VICE(I,J–1,1) – VICE(I–1,J,1) – VICE(I–1,J–1,1)),

where DELTAX is the x (longitude) grid spacing in degrees, DELTAY is the y (latitude) grid spacing in degrees, UICE is the x component of the ice drift, and VICE is the y component of the ice drift. E11, E22, and E12 are then redefined in the spherical coordinate system by:

E11(I,J) = E11(I,J) * DELTAX/DELTXA(J),

E22(I,J) = E22(I,J) * DELTAY/DELTYA,

$$E12(I,J) = (1.0/2.0/DELTYA) * (UICE(I,J,1) + UICE(I-1,J,1) - UICE(I,J-1,1) - UICE(I-1,J-1,1))/2.0 + (1.0/2.0/DELTXA) * (VICE(I,J,1) + VICE(I,J-1,1) - VICE(I-1,J,1) - VICE(I-1,J-1,1))/2.0,$$

where DELTXA and DELTYA are the x and y grid spacings, respectively, for thermodynamic fields.

After setting up the strain rates, the nonlinear bulk viscosity (ZETA) for each velocity field is calculated:

$$DELT = (E11(I,J) ** 2 + E22(I,J) ** 2) * (1.0 + ECM2) + 4.0 * ECM2 * E12(I,J) ** 2 + 2.0 * E11(I,J) * E22(I,J) * (1.0-ECM2),$$

$$DELT1 = sqrt(DELT),$$

$$DELT1 = MAX(GMIN, DELT1),$$

$$ZETA(I,J) = 0.5 * PRESS(I,J)/DELT1),$$

where PRESS is the ice strength. Note that ZETA cannot be less than 1.0×10^{-20}.

The minimum and maximum values of ZETA are set to:

$$ZMIN = 4.0 * 10^{8},$$

$$ZMAX = ((5.0 * 10^{12)}/(2.0 * 10^{4})) * PRESS(I,J).$$

The value of ZMAX is based on the pressure for each thermodynamic field. The value of ZETA must be within these minimum and maximum values.

The nonlinear shear viscosity (ETA) can then be calculated for each thermodynamic field:

$$ETA(I,J) = ECM2 * ZETA(I,J).$$

The strain rates are zeroed out at the land/sea boundary points for the thermodynamic fields:

$$E11(I,J) = E11(I,J) * HEFFM(I,J)$$

$$E22(I,J) = E22(I,J) * HEFFM(I,J)$$

$$E12(I,J) = E12(I,J) * HEFFM(I,J)$$

The strain rates are returned to the FORM CSU.

3.3.5 CSC Ice Thickness Distribution

The Ice Thickness Distribution CSC accounts for the changes in ice thickness and concentration due to growth, advection, and deformation of the ice. Only thick and thin ice are considered for these calculations. The deformation of thick ice can create thin ice by divergence while the thin ice can be removed by convergence. Both growth and melt affect the amount of thick and thin ice. While growth can significantly decrease the amount of thin ice and increase the amount of thick ice, melting can add to the amount of thin ice or create open water by decreasing the thick ice. The advection of the ice thickness and compactness due to explicit timestepping is done in the ADVECT CSU. The DIFFUS CSU is used to determine the diffusion of the ice thickness, compactness, and

concentration by using the explicit forward time differencing. The negative ice to be melted is calculated in the RNEGT CSU. The changes of thickness and compactness for each timestep are then estimated in the GROWTH CSU.

3.3.5.1 CSU ADVECT

The ADVECT CSU does the explicit timestepping for the advection of the ice thickness and compactness within the ice model.

3.3.5.1.1 CSU ADVECT Design Specification/Constraints – There are no known constraints.

3.3.5.1.2 CSU ADVECT Design

Input Data Elements:

DIFF1	harmonic diffusion constant
HEFF	mean ice thickness per grid cell or ice mass per grid area
HEFFM	thermodynamic land/sea mask
LAD	type of time finite difference (leapfrog or backward Euler)
UICEC	intermediate x component of the ice drift
VICEC	intermediate y component of the ice drift

/STEP/

DELTAT	timestep (s)
DELTAX	x (longitude) grid spacing (deg)
DELTAY	y (latitude) grid spacing (deg)

/STEPSP/

DELTXA	x (longitude) grid spacing for thermodynamic fields(m)
DELTYA	y (latitude) grid spacing for thermodynamic fields (m)

Output Data Elements:

HEFF	mean ice thickness per grid cell or ice mass per grid area

/DIFFU3/

DIFF3	harmonic diffusion constant

Parameters:
stored in file ice.par

The ice.par parameters are described in Sec. 3.1.1.2.

Data element design information is provided in Sec. 4.0.

Local Data Elements:

DELTT	timestep according to the type of time finite difference	real
DELTX	grid spacing in x direction	real
DELTY	grid spacing in y direction	real
DIFF2	harmonic diffusion constant	real
I	index counter	integer
J	index counter	integer
K3	index of time level	integer

KD	loop index for diffusion (2)	integer
LL	type of time finite difference	integer
RADIAN	degrees per radian (57.29578°)	real
RADIUS	radius of the Earth (6370.0×10^3 m)	real

Algorithms:

The advection terms are calculated by using the following:

$$\left[(uh)_x\right]_{i+1/2,j+1/2} = \frac{1}{4h}\left[\left(h_{i+1/2,j+1/2} + h_{i+3/2,j+1/2}\right)\left(u_{i+1,j+1} + u_{i+1,j}\right)\right.$$
$$\left. - \left(h_{i+1/2,j+1/2} + h_{i-1/2,j+1/2}\right)\left(u_{i-1,j+1} + u_{i-1,j}\right)\right], \qquad (18)$$

where h is the x (longitude) grid spacing and u is the x component of the ice-drift velocity. Solving for $(vh)_y$, where h is the y (latitude) grid spacing and v is the y component of the ice-drift velocity, is done similarly, but

$$-\tan(lat) * h * v/r \qquad (19)$$

is an extra term needed to transform from the Cartesian to spherical coordinates. *lat* is the latitude, h is the ice thickness or concentration, v is the ice drift along the y (latitude) axis, and r is the radius of the Earth.

Data Conversions:

The grid spacing, in degrees, of the latitudes must be converted to radians.

Logic Flow:

The ICEMDL CSU calls the ADVECT CSU.

Two types of time finite differences are allowed—backward Euler or leapfrog. For this version of the ice model, the backward Euler method is selected. For the backward Euler method, the timestep based on the type of time finite difference is set to the timestep (seconds) for the model run. If the leapfrog method were selected, the timestep based on the type of time finite difference would be two times the timestep (seconds) for the model run. The timestep index ($K3$) is set to 3 for the leapfrog method and 2 for the backward Euler method.

The mean ice thickness or ice mass per grid area (HEFF) is reordered so that the third time level is equal to the second time level and the second time level is equal to the first for each thermodynamic field:

HEFF(I,J,3) = HEFF(I,J,2)

HEFF(I,J,2) = HEFF(I,J,1)

Next, the process of calculating the standard conservative advection begins for each velocity field. First DELTX and DELTY are defined in the spherical coordinates with DELTX dependent on the latitude:

DELTX = DELTT/(4.0 * DELTAX),

DELTY = DELTT/(4.0 * DELTAY),

where DELTT is the timestep according to the type of time finite difference, DELTXA is the x (longitude) grid spacing in meters, DELTYA is the y (latitude) grid spacing in meters, and J is the index of the velocity field.

Then the advection term for all velocity fields at the first time level is calculated as:

HEFF(I+1,J+1,1) = HEFF(I+1,J+1,K3) – DELTX
* (HEFF(I+1,J+1,2) + HEFF(I+2,J+1,2)) * (UICEC(I+1,J+1) + UICEC(I+1,J))
– (HEFF(I+1,J+1,2) + HEFF(I,J+1,2)) * (UICEC(I,J+1) + UICEC(I,J)))
– DELTY * ((HEFF(I+1,J+1,2) + HEFF(I+1,J+2,2))
* (VICEC(I,J+1) + VICEC(I+1,J+1)) – (HEFF(I+1,J+1,2) + HEFF(I+1,J,2))
* (VICEC(I,J) + VICEC(I+1,J))),

where UICEC is the intermediate x component of the ice drift and VICEC is the intermediate y component of the ice drift. Due to the transformation to spherical coordinates from the Cartesian coordinates, the following extra term is added:

HEFF(I+1,J+1,1) = HEFF(I+1,J+1,1)
– TAN(DELTAY * (J–MIDY–0.5)/RADIAN) * HEFF(I+1,J+1,2)
* (VICEC(I,J) + VICEC(I+1,J) + VICEC(I,J+1) + VICEC(I+1,J+1))/4.0/RADIUS,

where DELTAY is the y (latitude) grid spacing in degrees, RADIAN is the constant to convert from degrees to radians, VICEC is the ice-drift velocity along the y axis, and RADIUS is the radius of the Earth in meters.

If the backward Euler time finite difference is selected, the value of HEFF at each velocity field is corrected. First, the value of HEFF at the third time level is set equal to the value of HEFF at the second time level and the value of HEFF at the second time level is set to be the average of the values at the first and second time levels:

HEFF(I,J,3) = HEFF(I,J,2)

HEFF(I,J,2) = 0.5 * (HEFF(I,J,1) + HEFF(I,J,2))

The values of LL = 3 and K3 = 3 are initialized.

The standard conservative advection process is repeated. After completion, the value of HEFF at the second time level is set to be the value of HEFF at the third time level:

HEFF(I,J,2) = HEFF(I,J,3)

Then HEFF is diffused at each velocity field for the third time level. Harmonic and biharmonic terms are computed. First, the value is set as:

DIFF3(J) = DIFF1,

where DIFF3 is the biharmonic diffusion constant at each y of the thermodynamic field and DIFF1 is the harmonic diffusion constant.

HEFF is recalculated at the first time level for each velocity field before calling the DIFFUS CSU. HEFF is redefined as:

HEFF(I,J,1) = HEFF(I,J,1) + DELTT * DIFF1 * HEFFM(I,J)
* (– TAN(DELTAY * (J–MIDY–0.5)/RADIAN))
* (HEFF(I,J+1,3) – HEFF(I,J–1,3))/2.0/DELTYA/RADIUS.

Then the DIFFUS CSU is called to determine the diffusion of ice thickness or concentration. After returning from the DIFFUS CSU, the value of HEFF is recalculated for the first time level at the thermodynamic land/sea mask as

HEFF(I,J,1) = (HEFF(I,J,1) + HEFF(I,J,3)) * HEFFM(I,J).

HEFF(I,J,3) from the previous call to the DIFFUS CSU now becomes the harmonic term. The values of the biharmonic term DIFF2 and the harmonic term (DIFF3) are defined in spherical coordinates:

DIFF2 = – DELTYA * DELTYA/DELTT

DIFF3(J) = DIFF2

Before executing the DIFFUS CSU, HEFF is redefined as:

HEFF(I,J,1) = HEFF(I,J,1) – HEFFM(I,J)
* (– TAN(DELTAY * (J–MIDY–0.5)/RADIAN))
* (HEFF(I,J+1,3) – HEFF(I,J–1,3))/2.0 * (DELTAY/RADIAN).

The DIFFUS CSU is called again to determine the diffusion of the ice thickness or concentration. After executing the DIFFUS CSU, the value of HEFF is recalculated for the first time level at the thermodynamic land/sea mask as:

HEFF(I,J,1) = (HEFF(I,J,1) + HEFF(I,J,3)) * HEFFM(I,J).

Upon completion of this CSU, flow returns to the main program.

3.3.5.2 CSU DIFFUS

The purpose of the DIFFUS CSU is to determine the diffusion of the ice thickness and concentration by using explicit forward time differencing.

3.3.5.2.1 CSU DIFFUS Design Specification/Constraints – There are no known constraints.

3.3.5.2.2 CSU DIFFUS Design

Input Data Elements:

DIFF1	harmonic diffusion constant
HEFF	mean ice thickness or concentration per grid cell on input; diffused value on output
HEFFM	thermodynamic land/sea mask

/DIFFU3/

DIFF3	harmonic diffusion constant

/STEP/

DELTAX	x (longitude) grid spacing (deg)
DELTAY	y (latitude) grid spacing (deg)

/STEPSP/

DELTXA	x (longitude) grid spacing for thermodynamic fields (m)
DELTYA	y (latitude) grid spacing for thermodynamic fields (m)

Output Data Elements:

HEFF	mean ice thickness or concentration per grid cell on input; diffused value on output

Parameters:
stored in file ice.par

The ice.par parameters are described in Sec. 3.1.1.2.

Data element design information is provided in Sec. 4.0.

Local Data Elements:

DELTXX	x grid spacing based on timestep, grid spacing, and diffusion constant	real
DELTYY	y grid spacing based on timestep, grid spacing, and diffusion constant	real
HEFF1(IMT,JMT)	diffused ice thickness or concentration	real
I	index counter	integer
J	index counter	integer

Logic Flow:

The DIFFUS CSU is called by the ADVECT CSU to determine the diffusion of ice thickness or concentration. All values of HEFF1 are initialized to 0. Then, DELTXX and DELTYY are calculated for each row of velocity fields.

DELTXX = DELTT * DIFF1/(DELTAX * * 2),

DELTYY = DELTT * DIFF1/(DELTAY * * 2),

where DELTT is the timestep, DIFF1 is the harmonic diffusion constant, DIFF3 is the diffusion constant, DELTXA is the x (longitude) grid spacing in degrees and DELTYA is the y (latitude) grid spacing in degrees. Now the diffused value of the ice thickness or concentration is calculated by:

$$\begin{aligned} HEFF1(I,J) = {} & DELTXX * ((HEFF(I+1,J,3) - HEFF(I,J,3)) * HEFFM(I+1,J) \\ & - (HEFF(I,J,3) - HEFF(I-1,J,3)) * HEFFM(I-1,J) \\ & + DELTYY * ((HEFF(I,J+1,3) - HEFF(I,J,3)) * HEFFM(I,J+1) \\ & - (HEFF(I,J,3) - HEFF(I,J-1,3)) * HEFFM(I,J-1)). \end{aligned}$$

The values of each thermodynamic field at the third time level is then set equal to the diffused value

$$HEFF(I,J,3) = HEFF1(I,J)$$

3.3.5.3 CSU GROWTH

The GROWTH CSU calculates the change of thickness and compactness for each timestep.

3.3.5.3.1 CSU GROWTH Design Specification/Constraints – There are no known constraints.

3.3.5.3.2 CSU GROWTH Design

Input Data Elements:

A22	minimum ice concentration
FO	growth rate of thin ice
HDIFF1	net growth of thin ice
HEFFM	land/sea mask for thermodynamic fields
HO	minimum ice thickness
OUT	land/sea mask including outflow conditions for thermodynamic variables

/GROW/

FHEFF	total growth rate of thick ice

/RSTRT/

AREA1	fraction of grid cell covered by ice
HEFF	mean ice thickness per grid cell

/STEP/

DELTAT	timestep (s)

Output Data Elements:

FO	growth rate of thin ice
GAREA	change of areal ice extent due to melting and freezing
HCORR	additional ice to be melted for the mixed-layer balance
HDIFF1	net growth of thin ice

/GROW/

FHEFF	total growth rate of thick ice

/RSTRT/

AREA1	fraction of grid cell covered by ice
HEFF	mean ice thickness per grid cell

Parameters:
stored in file ice.par

The ice.par parameters are described in Sec. 3.1.1.2.

Data element design information is provided in Sec. 4.0.

Local Data Elements:

GHEFF(IMT,JMT)	average growth tendency for the timestep	real
I	index counter	integer
J	index counter	integer

Logic Flow:

The GROWTH CSU is called by the ICEMDL CSU.

Upon entering this CSU, the growth rates of thick and thin ice are used to calculate the total change of ice thickness and concentration in a grid cell. To do this, the average growth tendency of this timestep (GHEFF), the average compactness tendency for this timestep (GAREA), the net growth of thin ice for this timestep (HDIFF1), and the additional ice to be melted for the mixed-layer balance (HCORR) are calculated first:

$$GHEFF(I,J) = -DELTAT * FHEFF(I,J),$$

$$GAREA(I,J) = DELTAT * FO(I,J),$$

$$GHEFF(I,J) = -1.0 * MIN(HEFF(I,J,1),\ GHEFF(I,J)),$$

$$HDIFF1(I,J) = -DELTAT * HDIFF1(I,J),$$

$$GAREA(I,J) = MAX(0.0,\ GAREA(I,J),$$

$$HDIFF1(I,J) = -1.0 * MIN(HEFF(I,J,1),\ HDIFF1(I,J)),$$

$$HCORR(I,J) = MIN(0.0,\ GHEFF(I,J)),$$

where DELTAT is the timestep in seconds, FHEFF is total growth rate of thick ice, FO is the growth rate of thin ice, and HEFF is the mean ice thickness per grid cell.

The value of GAREA must not be less than 0. Then the change of areal ice extent due to melting and freezing is calculated by:

$$GAREA(I,J) = 2.0 * (1.0-AREA1(I,J,2)) * GAREA(I,J)/HO + 0.5 * HCORR(I,J) * AREA1(I,J,2)/HEFF(I,J,1) + 0.00001),$$

where AREA1 is the fraction of the grid cell covered by ice and HO is the demarcation between thick and thin ice. The values of AREA1 and HEFF are corrected by:

$$AREA1(I,J,1) = AREA1(I,J,1) + GAREA(I,J),$$

$$HEFF(I,J,1) = HEFF(I,J,1) + GHEFF(I,J) * OUT(I,J),$$

where OUT is the land/sea mask including outflow conditions for thermodynamic variables. The value of HCORR is recalculated by:

$$HCORR(I,J) = GHEFF(I,J) - DELTAT * FHEFF(I,J).$$

The RNEGT CSU is called to remove the "negative" ice from the ice thickness field.

The outside points in the AREA1 and HEFF arrays are zeroed out by multiplying the current value by the land/sea mask value for each point

$$AREA1(I,J,1) = AREA1(I,J,1) * HEFFM(I,J)$$

$$HEFF(I,J,1) = HEFF(I,J,1) * HEFFM(I,J).$$

If there is no ice at a point, then the value of AREA1 is set to 0. This new value of AREA1 is checked again to make sure that it is not greater than 1 and is not less than the minimum

compactness allowed. The amount of additional ice to be melted for the mixed-layer balance (HCORR) is calculated by:

$$HCORR(I,J) = GHEFF(I,J) - DELTAT * FHEFF(I,J).$$

The values of total growth rate of the thick ice (FHEFF) and the growth rate of thin ice (FO) are stored.

The ice thickness, ice concentration, net growth rate of thin ice, growth rate of thin ice, the negative ice to be melted, and the total growth rate of thick and thin ice are returned to the main driving CSU.

3.3.5.4 CSU RNEGT

The RNEGT CSU removes the negative ice from the model. This method essentially removes the amount of thin ice due to converging conditions.

3.3.5.4.1 CSU RNEGT Design Specification/Constraints – There are no known constraints.

3.3.5.4.2 CSU RNEGT Design

Input Data Elements:

FH	additional ice to be melted
HEFF	mean ice thickness per grid cell

Output Data Elements:

HEFF	mean ice thickness per grid cell

Parameters:
stored in file ice.par

The ice.par parameters are described in Sec. 3.1.1.2.

Data element design information is provided in Sec. 4.0.

Local Data Elements:

I	index counter	integer
J	index counter	integer

Logic Flow:

The RNEGT CSU is called by the GROWTH CSU.

The melt value is subtracted from the ice thickness value at the first time level (HEFF) and assigned to the ice thickness value at the third time level:

$$HEFF(I,J,3) = HEFF(I,J,1) - FH(I,J).$$

The ice thickness value is tested to make certain that it is not less than 0. The new value of HEFF is returned to the GROWTH CSU.

3.3.6 CSC Ice Strength

The ice strength, as a function of ice thickness distribution, is calculated in this CSU. The strength of the ice depends on the amount of thin ice and is calculated in the FORM CSU.

3.3.6.1 CSU XSUM

The purpose of the XSUM CSU is to sum every value of the input array into a scalar.

3.3.6.1.1 CSU XSUM Design Specification/Constraints – There are no known constraints.

3.3.6.1.2 CSU XSUM Design

Input Data Elements:

HEFF	mean ice thickness per grid cell

Output Data Elements:

S1	summation of mean ice thickness values

Parameters:
stored in file ice.par

The ice.par parameters are described in Sec. 3.1.1.2.

Data element design information is provided in Sec. 4.0.

Local Data Elements:

I	index counter	integer
J	index counter	integer

Algorithms:

When calculating the ice thickness *SUM*, use

$$SUM = \sum_{j=1}^{ny} \sum_{i=1}^{nx} HEFF_{i,j} \ , \tag{20}$$

where *HEFF* is the ice thickness per grid cell, *ny* is the maximum number of columns of thermodynami fields allowed, and *nx* is the maximum number of rows of thermodynamic fields allowed.

Logic Flow:

CSU XSUM is called by the ICEMDL CSU to find the total ice in the basin.

The total basin ice thickness is calculated by summing the mean ice thickness for all thermodynami fields.

The summation of the mean ice thickness values is returned to the main driving program.

3.3.6.2 CSU FORM

See Sec. 3.3.3.1 for a detailed description of the FORM CSU.

3.3.7 CSC Heat Balance

The Heat Balance CSC is used to calculate the heat budget. The HEAT CSU calculates the terms needed and the BUDGET CSU uses these terms to compute the growth rates of thick and thin ice.

3.3.7.1 CSU HEAT

The HEAT CSU takes the FNMOC forcing and calculates terms needed for the heat budget balance.

3.3.7.1.1 CSU HEAT Design Specification/Constraints – There are no known constraints.

3.3.7.1.2 CSU HEAT Design

Input Data Elements:

FO	growth rate of ice on open water

/COX2/

CFO	either the heat above the freezing temperature in terms of ice thickness growth rate or the ice thickness growth rate in open water
TFRZ	temperature at the freezing point (Kelvin)
TM1	mixed-layer temperature from the previous timestep (Kelvin)

/OCEANS/

FW	oceanic heat flux

/RAD/

FSH	atmospheric forcing – solar radiation

/RFOR/

ES	NOGAPS atmospheric forcing – surface vapor pressure
ES1	NOGAPS atmospheric forcing – sensible heat flux
PS	NOGAPS atmospheric forcing – surface air pressure
PS1	NOGAPS atmospheric forcing – total heat flux
TAIR	NOGAPS atmospheric forcing – surface air temperature

/RFOR2/

GAIRX	x component of the geostrophic wind
GAIRY	y component of the geostrophic wind

/RSTRT/

AREA1	fraction of the grid cell covered by ice
HEFF	mean ice thickness per grid cell
TICE	mixed-layer temperature for open water or ice temperature for ice-covered water

Output Data Elements:

FO	growth rate of ice on open water
HDIFF1	net growth of thin ice
SHICE	total ice thickness

/COX2/

CFO	either the heat above the freezing in terms of ice thickness growth rate or the ice thickness growth rate in open water

FW1	heat above the freezing temperature
GICE	ice thickness growth rate of open water

/GROW/

FHEFF	total growth rate of thick ice

/RFOR/

TAIR	NOGAPS atmospheric forcing – surface air temperature

/RSTRT/

AREA1	fraction of the grid cell covered by ice
TICE	mixed-layer temperature for open water or ice temperature for ice-covered water

Parameters:
stored in file ice.par

The ice.par parameters are described in Sec. 3.1.1.2.

Data element design information is provided in Sec. 4.0.

Local Data Elements:

A22	minimum ice concentration	real
AR(IMT,JMT)	used in determining the amount of open area	real
CFO_TMP	temporary storage for heat flux	real
CP	conversion factor from joules to calories (4.19×10^6)	real
EXCHNG	frequency of exchanging ice/ocean data	real
FLO(IMTP1,JMTP1)	net longwave radiation	real
GX	x component of the geostrophic wind speed	real
GY	y component of the geostrophic wind speed	real
HICE(IMT,JMT)	average ice thickness over the grid cell	real
I	index counter	integer
IMT	total number of T grid boxes zonally (360)	integer
J	index counter	integer
JMT	total number of T grid boxes meridionally (360)	integer
KOPEN	flag for open water or ice-covered water calculations	integer
QA(IMTP1,JMTP1)	specific humidity at the ice surface	real
RLATNT	volumetric heat of fusion of ice (302×10^6 joules)	real
TDAY	timestep to exchange ice-ocean conditions	real
TMIX(IMT,JMT)	mixed-layer temperature	real
UG(IMT,JMT)	magnitude of wind	real
ZMIX	mixed-layer thickness (30 m)	real

Algorithms:

The surface heat budget (Parkinson and Washington 1979; Manabe et al. 1979) is calculated using the following equation:

$$(1-\alpha)F_s + F_L + D_1\left|\vec{U_g}\right|(T_a - T_0) + D_2\left|\vec{U_g}\right|[q_a(T_a) - q_s(T_0)] - D_3 T_0^4 + (K/H)(T_w - T_0) = 0\,, \quad (21)$$

where α is the surface albedo, T_0 is the surface temperature of ice, T_a is the air temperature, T_w is the water temperature, $\left|\vec{U_g}\right|$ is the geostrophic wind, q_a is the specific humidity of air, q_s is the

specific humidity of the ice surface, F_s is the incoming short-wave radiation, F_L is the incoming longwave radiation, D_1 is the bulk sensible heat transfer coefficient, D_2 is the bulk latent heat transfer coefficient (water or ice), D_3 is the Stefan-Boltzmann constant times the surface emissivity, K is the ice conductivity, and H is the ice thickness.

Logic Flow:

The HEAT CSU is called by the ICEMDL CSU.

To begin the process of solving the equation for the surface heat budget, the x (GX) and y (GY) components of geostrophic wind speed are calculated for each thermodynamic field. These x and y components, which are an average of the wind speed at the corners of the grid cell, are then used to find the magnitude of the geostrophic wind (UG):

GX = (GAIRX(*I*,*J*) + GAIRX(I+1,J) + GAIRX(*I*,*J*+1) + GAIRX(I+1,J+1)) * 0.25

GY = (GAIRY(*I*,*J*) + GAIRY(I+1,J) + GAIRY(*I*,*J*+1) + GAIRY(I+1,J+1)) * 0.25

UG(*I*,*J*) = sqrt(GX * * 2 + GY * * 2).

Next, the specific humidity at the ice surface (QA) for each thermodynamic field is calculated from the surface vapor pressure and the surface pressure fields of atmospheric forcing.

QA(*I*,*J*) = (0.622 * ES(*I*,*J*))/(PS(*I*,*J*) – ES(*I*,*J*)),

where ES is the surface vapor pressure and PS is the surface air pressure.

The net long wave radiation (FLO) is calculated using the total heat flux minus the sensible heat flux plus the solar radiation. The sign of the net long wave radiation is then changed to correct the net long wave error.

FLO(*I*,*J*) = PS1(*I*,*J*) – ES1(*I*,*J*) + FSH(*I*,*J*)

FLO(*I*,*J*) = – FLO(*I*,*J*),

where PS1 is the total heat flux, ES1 is the sensible heat flux and FSH is the solar radiation.

The area of the grid cell covered by thick ice cannot be <0.15; each value at the second time level is checked to make sure this does not happen.

Next, the total growth rate of thick ice (FHEFF) is set to 0, the average ice thickness over the grid cell (HICE) is calculated, and the term to be used in determining the amount of ice (AR) is determined for each thermodynamic field:

FHEFF(*I*,*J*) = 0,

HICE(*I*,*J*) = HEFF(*I*,*J*,2)/AREA1(*I*,*J*,2),

AR(*I*,*J*) = MIN(AREA1(*I*,*J*,2), HEFF(*I*,*J*,2) * 10000).

Constant values are assigned for the volumetric heat of fusion (RLATNT), the conversion factor from joules to calories (CP), and the thickness of the mixed layer (ZMIX). The exchange of

ice and ocean conditions (EXCHNG) is set to every 3 h or 8 times per day, and the number of seconds within this exchange time (TDAY) is also set:

$$RLATNT = 302.0 * 10^6$$

$$CP = 4.19 * 10^6$$

$$ZMIX = 30.0$$

$$EXCHNG = 8.0$$

$$TDAY = 86400.0/EXCHNG.$$

The value of TMIX is reset to the mixed-layer temperature (Kelvin) from the previous timestep for each thermodynamic field:

$$TMIX(I,J) = TM1(I,J).$$

The value of KOPEN is set to –1 for the open water case and the BUDGET CSU is called to calculate the change in growth rate for open water.

Next, the value of KOPEN is set to 2 for an ice-covered ocean case and the BUDGET CSU is called again, but this time it calculates the change in growth rate for ice-covered water.

After executing from the BUDGET CSU for the second time, the fractional rate of total open water growth (HDIFF1) per grid cell, the net rate of total open water growth rate (FW1), the heat flux (CFO_TMP), and the heat above the freezing in terms of ice thickness growth rate (CFO) are calculated:

$$HDIFF1(I,J) = (1.0 - AR(I,J)) * FO(I,J)$$

$$FW1(I,J) = FO(I,J)$$

$$CFO\_TMP = CP * ZMIX * (TM1(I,J)-TFRZ(I,J))/TDAY/RLATNT$$

$$CFO(I,J) = CFO\_TMP * (1.0-AR(I,J)).$$

After these values are calculated, quality checks are performed. If the calculated value of CFO is <0, then it is set to 0. If the calculated value of the ice thickness growth rate HDIFF1 <0, then HDIFF1 is not changed since the ice thickness has not increased. The ice is considered to have melted for this case. If HDIFF1 ≥0 and HDIFF1 ≥CFO, then the ice grows in the open water. If HDIFF1 ≥0 and HDIFF1 <CFO, then the ice is stable; i.e., the ice is neither grown nor melted. For this case, the water is cooled but the ice does not grow because the atmospheric cooling cannot take away all of the heat above the freezing temperature. HDIFF1 and FW1 are reset to 0, i.e., there are no ice growth rates.

If (HDIFF(I,J) ≥0.0), then
  if (HDIFF1(I,J) ≥CFO(I,J)), then
    HDIFF1(I,J) = HDIFF1(I,J) – CFO(I,J)
    FW1(I,J) = FO(I,J) – CFO_TMP,
  else
    HDIFF1(I,J) = 0
    FW1(I,J) = 0.

The ice thickness growth rate of open water and the total ice thickness are calculated by:

$GICE(I,J) = (1.0 - AR(I,J)) * (FO(I,J) + FW(I,J) * 10^{-6}/302.0)$,

$SHICE(I,J) = FHEFF(I,J) * AR(I,J) + HDIFF1(I,J)$.

The values of FHEFF, FO, and FW1 are redefined as:

$CFO(I,J) = (1.0 - AR(I,J)) * FO(I,J)$,

$FHEFF(I,J) = FHEFF(I,J) * AR(I,J) + HDIFF(I,J)$,

$FO(I,J) = FW1(I,J)$,

$FW1(I,J) = CP * ZMIX * (TM1(I,J) - TFRZ(I,J)) * (1.0 - AR(I,J))/TDAY/RLATNT$,

$FW1(I,J) = MAX(0, FW1(I,J))$.

The program flow returns to the ICEMDL CSU.

3.3.7.2 CSU BUDGET

The purpose of the BUDGET CSU is to compute the growth rates of thick and thin ice. It also computes the surface temperature of ice by iteration. This temperature balances the surface heat budget and dictates the conduction of heat through the ice and, therefore, the growth rates.

3.3.7.2.1 CSU BUDGET Design Specification/Constraints – There are no known constraints.

3.3.7.2.2 CSU BUDGET Design

Input Data Elements:

FLO	longwave radiation
HICE1	average ice thickness over the grid cell
KOPEN	flag for ice-covered or open water
QA	specific humidity at the ice surface
TAIR	NOGAPS atmospheric forcing – surface air temperature
TSFC	mixed-layer temperature for open water or ice temperature for ice-covered water (Kelvin)
UG	magnitude of the wind

/OCEANS/

FW	oceanic heat flux

/RAD/

FSH	atmospheric forcing – solar radiation

/SNOW/

SNRT	snowfall rate

/STEP/

DELTAT	timestep (s)

Output Data Elements:

TSFC	mixed-layer temperature for open water or ice temperature for ice-covered water (Kelvin)
TSUM	mean ice thickness from all seven levels

Parameters:
stored in file ice.par

The ice.par parameters are described in Sec. 3.1.1.2.

Data element design information is provided in Sec. 4.0.

Local Data Elements:

A1(IMT,JMT)	fixed forcing term in heat budget	real
A2(IMT,JMT)	used in heat budget	real
A3(IMT,JMT)	used in heat budget	real
AKI	ice conductivity constant for the ice cover case (2.1656 W $m^{-1}K^{-1}$)	real
AKS	snow conductivity constant for the ice cover case (0.31)	real
B(IMT,JMT)	used in heat budget	real
D1	bulk sensible heat transfer coefficient (2.284 j/m^3 K)	real
D1I	bulk latent heat transfer coefficient over ice (6.4475×10^3 j/m^3)	real
D1W	bulk latent heat transfer coefficient over water (5.6875×10^3 j/m^3)	real
D3	Stefan-Boltzmann constant times the surface emissivity (5.5×10^{-8} W/m^2 K^4)	real
EM(IMT,JMT)	used in computing terms in the heat budget for an ice-covered case	real
FICE(IMT,JMT)	ice growth at each of the seven levels	real
FW2(IMT,JMT)	oceanic heat flux (W/m^2)	real
HICE(IMT,JMT)	ice depth at each of the seven levels	real
HSI(IMT,JMT)	snow depth at each of the seven levels	real
I	index counter	integer
IMAX	maximum number of iterations (10)	integer
ITER	counter for iterations	integer
J	index counter	integer
NLEVEL	number of levels (7)	integer
NLV	index of loop over number of levels	integer
NZ3	used to find the middle level	integer
POOL1(IMT)	temporary array	real
POOL2(IMT)	temporary array	real
Q0(IMT,JMT)	the inverse of volumetric heat of fusion of ice (m^3/j)	real
QIS	used in equation for determining excess heat after all snow is melted (0.364)	real
QS1	used to calculate terms in heat budget (6.1402×10^{-4})	real
SNOW(IMT,JMT)	stores midpoint value of snow depth at all levels	real
SSUM(IMT,JMT)	mean of snow depth from all seven levels	real
TB	freezing point of seawater (271.2 K)	real
THICK(IMT,JMT)	midpoint value of ice thickness	real
TICE(IMT,JMT)	temperature of mixed layer for open water case; ice temperature for ice-covered case	real
TMELT	freezing point (273.16 K)	real
TMID(IMT,JMT)	midpoint surface temperature	real
TMID2(IMT,JMT)	midpoint surface temperature of the next timestep	real
TX1	ice constant	real
TX2	snow constant	real
TY	used in computing snow and ice thickness	real
TY1	ice constant	real
TZ	surface ice temperature at each level (°C)	real

TZ1	ice constant	real
TZ2	snow constant	real
V1	used to correct the snow depth	real

Algorithms:

The thermal conductivity is a single value based on a weighted sum of snow and ice conductivities:

$$\frac{K_s K_I}{\left(K_s\, Sn_{LVL} + K_I h_{LVL}\right)}, \tag{22}$$

where K_s is the snow conductivity, K_I is the ice conductivity, Sn_{LVL} is the snow depth at the current level, and h_{LVL} is the ice thickness at the same level.

The surface heat budget (Parkinson and Washington 1979; Manabe et al. 1979) is calculated by using:

$$\begin{aligned}(1-\alpha)\,F_s + F_L + D_1\left|\vec{U}_g\right|(T_a - T_0) + D_2\left|\vec{U}_g\right|[q_a(T_a) - q_s(T_0)] - D_3\,T_0^4 \\ + (K/H)(T_w - T_0) = 0\,,\end{aligned} \tag{23}$$

where α is the surface albedo, T_0 is the surface temperature of ice, T_w is the air temperature, T_a is the water temperature, $\vec{U}_g$ is the geostrophic wind, q_a is the specific humidity of air, q_s is the specific humidity of the ice surface, F_s is the incoming short-wave radiation, F_L is the incoming longwave radiation, D_1 is the bulk sensible heat transfer coefficient, D_2 is the bulk latent heat transfer coefficient (water or ice), D_3 is the Stefan-Boltzmann constant times the surface emissivity, K is the ice conductivity, and H is the ice thickness.

Logic Flow:

The HEAT CSU calls the BUDGET CSU.

In BUDGET CSU, the constant QS1, the freezing point of seawater TB, the maximum number of iterations IMAX, the bulk heat transfer coefficient D1, the bulk latent heat transfer coefficient over water D1W, the bulk latent heat transfer coefficient over ice D1I, the Stefan-Boltzmann constant D3, and the freezing point TMELT are defined:

QS1 = 0.622/1013.0

IMAX = 10.0

TB = 271.2

D1 = 2.284

D1W = $5.6875 * 10^3$

D1I = $6.4475 * 10^3$

D3 = $5.5 * 10^{-8}$

TMELT = 273.16.

The initial snow depth HSI is set to 0 at each of the thermodynamic fields.

The oceanic heat flux is computed for each thermodynamic field as

$$FW2(I,J) = FW(I,J).$$

For the ice-covered case, some initial conditions are assigned at each thermodynamic field:

$$EM(I,J) = 0.99$$
$$HICE(I,J) = 0.80$$
$$POOL1(I) = 0.97$$
$$Q0(I,J) = (1.0 * 10^{-6})/110.0,$$

where EM, HICE, and POOL1 are all values used in computing the heat budget, and Q0 is the inverse of the volumetric heat of fusion of ice. Next, the value of the snow depth (HSI) is checked and the value of EM is reassigned accordingly:

$$\text{if } (HSI(I,J) = 0), \text{ then set } EM(I,J) = POOL1(I).$$

Each value of POOL1 is reassigned, the value of HICE is checked and the value of Q0 is reassigned if the snow depth (HSI) is equal to 0.

$$POOL1(I) = (1.0 * 10^{-6})/302.0$$
$$HICE(I,J) = MAX(HICE1(I,J),\ 0.05)$$
$$\text{if } (HSI(I,J) = 0), \text{ then set } Q0(I,J) = POOL1(I).$$

Next, the fixed forcing term in the heat budget can be determined for the case of ice cover:

$$A1(I,J) = FSH(I,J) + FLO(I,J) + D1 * UG(I,J) * TAIR(I,J) + D1I * UG(I,J) * QA(I,J),$$

where FSH is the solar radiation, UG is the geostrophic wind, TAIR is the surface air temperature and QA is the specific humidity at the ice surface.

For the ice-covered case, the growth at the seven levels is calculated after some initial conditions are set up at each thermodynamic field for the midpoint value of ice thickness THICK, the midpoint value of snow depth SNOW, the mean ice thickness TSUM, the mean snow depth SSUM, and the midpoint surface temperature TMID:

$$THICK(I,J) = HICE(I,J)$$
$$SNOW(I,J) = HSI(I,J)$$
$$TSUM(I,J) = 0$$
$$SSUM(I,J) = 0$$
$$TMID(I,J) = TSFC(I,J).$$

Now the snow and ice thickness (HSI and HICE, respectively) are calculated for the current level at each thermodynamic field:

$$TSFC(I,J) = TMID(I,J)$$
$$HICE(I,J) = NLV * 2.0 * THICK(I,J)/(NLEVEL + 1)$$

TZ = TSFC(I,J) – 273.16

TY = NLV * 2.0 * SNOW(I,J)/(NLEVEL+1),

where NLV is the index of the current level and NLEVEL is the total number of levels to be processed. The value of HSI is reassigned based on the value of TZ:

if (TZ < 0) then HSI(I,J) = TY

if (TZ ≥ 0) then HSI(I,J) = SNOW(I,J).

The terms in the heat budget are computed using an iterative process. The iteration is started by calculating the following for each x and y thermodynamic field:

B(I,J) = QS1 * 6.11 * exp(21.8746 * (TSFC(I,J) – TMELT)/
(TSFC(I,J) – TMELT + 265.5))

A3(I,J) = D1I * UG(I,J) * B(I,J) * 21.8746 * 265.5/
((TSFC(I,J)–TMELT+265.5) * * 2)

A2(I,J) = – D1 * UG(I,J) * TSFC(I,J) – D1I * UG(I,J) * B(I,J)

TICE(I,J) = AKI * AKS/(AKS * HICE(I,J) +AKI * HSI(I,J))

A3(I,J) = A3(I,J) + 4.0 * D3 * EM(I,J) * TSFC(I,J) * * 3 + TICE(I,J)
+ D1 * UG(I,J)

B(I,J) = TICE(I,J) * (TB – TSFC(I,J)).

If the maximum number of iterations has not been reached, then the surface temperature of ice TSFC is determined at each x and y thermodynamic field:

TSFC(I,J) = TSFC(I,J) + (A1(I,J) + A2(I,J) + B(I,J))/A3(I,J),

If (TSFC(I,J) ≤ 200.0 and TAIR(I,J) ≥ TMELT),
then TSFC(I,J) = TMELT

If (TSFC(I,J) ≤ 200.0 and TAIR(I,J) < TMELT)
then TSFC(I,J) = 200.0.

The iteration counter is incremented and then checked to see if the maximum number of iterations has been reached. If it has, any value of the surface temperature of ice that is above freezing is reset to the freezing point (273.16 K). The iterative process continues until the maximum number of iterations has been computed.

After the maximum number of iterations has been computed, the ice temperature (TICE) at each x and y thermodynamic field is calculated:

TICE(I,J) = TICE(I,J) * (HICE(I,J)/AKI * TSFC(I,J) + HSI(I,J)/AKS * TB)

If |HSI| < 0.0001, it is set to 0.0001.

The depth of melted snow (POOL1) and the excess heat after all snow is melted (POOL2) are then calculated.

$$POOL1(I) = Q0(I,J) * (A1(I,J)+A2(I,J)+AKS/HSI(I,J) * (TICE(I,J)-TSFC(I,J)))$$

$$\begin{aligned} POOL2(I) = {} & (HSI(I,J)/DELTAT - POOL1(I)) * QIS \\ & + 1.0 * 10^{-6}/302.0 * (AKS/HSI(I,J) * (TICE(I,J)-TSFC(I,J)) \\ & - AKI/HICE(I,J) * (TB-TICE(I,J)) \\ & + 0.09 * FSH(I,J) + 0.02 * FLO(I,J)). \end{aligned}$$

If snow existed and POOL2 <0, then the ice is melted. If no snow existed, POOL1 is recalculated using ice constants, instead of snow constants, to obtain the ice melt:

$$TZ1 = HSI(I,J) - 0.0001$$

$$\begin{aligned} TY1 = {} & - POOL1(I) - (AKI/HICE(I,J) * (TB-TICE(I,J)) \\ & - AKS/HSI(I,J) * TICE(I,J)-TSFC(I,J))) * Q0(I,J) \end{aligned}$$

$$TX1 = MIN(0.0, POOL2(I))$$

if (TZ1 = 0), then $FICE(I,J) = TY1$

if (TZ1 ≠ 0), then $FICE(I,J) = TX1$

if (TZ1 = 0), then $HSI(I,J) = 0.$

The snow depth is corrected by:

$$V1 = HSI(I,J) - POOL(I) * DELTAT$$

$$POOL1(I) = MAX(V1, 0.0)$$

$$TZ2 = TSFC(I,J) - TMELT$$

$$TX2 = HSI(I,J) + DELTAT * SNRT$$

if (TZ2 = 0), then $HSI(I,J) = POOL1(I)$

if (TZ2 ≠ 0), then $HSI(I,J) = TX2.$

The ice melt at the ice-ocean interface is added to the melt at the top of the ice by:

$$FICE(I,J) = 1.0 * 10^{-6}/302.0 * (AKI/HICE(I,J) * (TB - TICE(I,J)) - FW2(I,J)) + FICE(I,J).$$

The snow and ice depth changes are added to all seven levels of the sum and the surface temperature of the ice is saved as the midpoint (TMID2) for the next timestep computations:

$$TSUM(I,J) = FICE(I,J)/NLEVEL + TSUM(I,J)$$

$$SSUM(I,J) = HSI(I,J)/NLEVEL + SSUM(I,J)$$

$$NZ3 = NLV - (NLEVEL+1)/2$$

if (NZ3 = 0) then $TMID2(I,J) = TSFC(I,J).$

After values for all seven levels are computed, then the following calculations made:

$$HSI(I,J) = SSUM(I,J)$$

$$TSFC(I,J) = TMID2(I,J).$$

This completes all computation for the ice-covered case.

For the open water case, the following are calculated at each x and y thermodynamic field:

$TICE(I,J) = TSFC(I,J)$

$Q0(I,J) = 1.0 * 10^{-6}/302.0$

$A1(I,J) = FSH(I,J) + FLO(I,J) + D1 * UG(I,J) * TAIR(I,J) + D1W * UG(I,J) * QA(I,J)$

$B(I,J) = QS1 * 6.11 * \exp(17.2694 * (TSFC(I,J) - TMELT)/(TSFC(I,J) - TMELT + 237.3))$

$A2(I,J) = - D1 * UG(I,J) * TSFC(I,J) - D1W * UG(I,J) * B(I,J)$

$TSUM(I,J) = Q0(I,J) * (-FW2(I,J) - A1(I,J) - A2(I,J) + SNRT * 110.0 * 10^6)$.

In the equations above, TICE is the temperature of the mixed layer.

The mean ice thickness from all seven levels (TSUM) and the mixed-layer temperature for the open water case or the ice temperature for ice-covered water (TSFC) are returned to the HEAT CSU.

3.4 CSC Compute Ocean

3.4.1 CSC Compute Ocean Driver

3.4.1.1 CSU OCEAN

CSU OCEAN serves as the main routine for the CSC Compute Ocean. This routine originated as the main routine in the stand-alone version of the Cox Ocean Model (Cox 1984). Minor modifications were made to allow the original routine to function as a subroutine in the CSCI PIPS2.0.

3.4.1.1.1 CSU OCEAN Design Specification/Constraints – There are no known constraints.

3.4.1.1.2 CSU OCEAN Design

Input Data Elements:

AREA	surface area of the ocean basin, read in on a restart or computed on a run started from scratch
FINS	floating point array of vorticity starting indices read in from the ocean restart file
HR	reciprocal depth array read in for model initialization
ITT	timestep counter read in on a restart
P	stream function computed in previous model execution for the last timestep processed; read in on a restart
PB	stream function computed in previous model execution for the second-to-last timestep processed; read in on a restart
VOLUME	volume of the ocean basin, read in on a restart or computed on a run started from scratch

/CORSP/

FCORSP	array of sine of latitude positions for each gridpoint

/LEVITUS/

SMIX	interpolated monthly salinity data input from the Levitus salinity file
TMIX	interpolated monthly temperature data input from Levitus temperature file

Output Data Elements:

/FIELDS/

HR	reciprocal depth array; read in on a restart and computed on a run started from scratch
P	mass transport stream function output to CSU OSTEP to initialize the timestepping
PB	mass transport stream function output to CSU OSTEP to initialize the timestepping
ZTD	array values are initialized to 0

/FULLWD/

AREA	surface area of the ocean basin, read in on a restart or computed on a run started from scratch
EB	sets the type of mixing timestep. If true, a Euler backward timestep is done. If false, a forward step is done.
IEIS	array of ending *I* indices for island boxes; values are initialized in a data statement; = 283, 138, 224, 227
IEZ	array of ending indices for vorticity
ISIS	array of starting *I* indices for island boxes; values are initialized in a data statement; = 272, 133, 220, 225
ISZ	array of starting indices for vorticity
ITT	timestep counter
JEIS	array of ending *J* indices for island boxes; values are initialized in a data statement; = 126, 173, 187, 182
JSIS	array if starting *J* indices for island boxes; values are initialized in a data statement; = 113, 168, 182, 179
KAR	used to enable vectorization
KFLDS	disk unit number for two-dimensional horizontal fields and start and end indices for vorticity; = 12
KONTRL	disk unit number for timestep counter; = 11
LABS(3)	array containing disk unit numbers for slabs; = 13, 14, and 15
MXSCAN	maximum number of scans allowed for convergence in CSU CRELAX initialized in a DATA statement; = 100
NA	initialized in a DATA statement; = 1
NB	initialized to 0
NC	initialized to 0
NDISK	permuting disk number for file containing slab data
NDISKA	permuting disk number for file containing slab data
NDW	initialized to 1,000,000
NMIX	initialized in a DATA statement; = 6
NTSI	initialized in a DATA statement; = 1
NWRITE	initialized to 1,000,000
TTSEC	initialized to 0
VOLUME	volume of the ocean basin, read in on a restart or computed on a run started from scratch

/ONEDIM/

C2DZ	vertical grid spacing times 2; DZ * 2
CS	cosine of U,V point latitudes
CSR	1.0/CS
CST	cosine of T point latitudes
CSTR	1.0/CST

DXT	zonal grid spacing across T boxes (between U,V points)
DXT2R	1.0/(DXT * 2.0)
DXT4R	1.0/(DXT * 4.0)
DXTR	1.0/DXT
DXU	zonal grid spacing across U,V boxes (between T points)
DXU2R	1.0/(DXU * 2.0)
DXU4R	1.0/(DXU * 4.0)
DXUR	1.0/DXU
DYT	meridional grid spacing across T boxes (between U,V points)
DYT2R	1.0/(DYT * 2.0)
DYT4R	1.0/(DYT * 4.0)
DYTR	1.0/DYT
DYU	meridional grid spacing across U,V boxes (between T points)
DYU2R	1.0/(DYU * 2.0)
DYU4R	1.0/(DYU * 4.0)
DYUR	1.0/DYU
DZ(15)	vertical grid spacing in centimeters of U,V,T boxes (between W points): DZ(1) = 3000.0 DZ(2) = 4630.0 DZ(3) = 6745.0 DZ(4) = 9446.0 DZ(5) = 12846.0 DZ(6) = 17057.0 DZ(7) = 22191.0 DZ(8) = 28351.0 DZ(9) = 35621.0 DZ(10) = 44057.0 DZ(11) = 53678.0 DZ(12) = 64454.0 DZ(13) = 76300.0 DZ(14) = 89071.0 DZ(15) = 102553.0
DZ2R	1.0/(DZ * 2.0)
DZZ	vertical grid spacing across W boxes (between U,V,T points)
DZZ2R	1.0/(DZZ * 2.0)
EEH	upper vertical mixing coefficient of T
EEM	upper vertical mixing coefficient of U,V
FFH	lower vertical mixing coefficient of T
FFM	lower vertical mixing coefficient of U,V
PHI	latitude in radians of the U,V points
PHIT	latitude in radians of the T points
SFU	external mode component of U
SFUB	external mode component of U for the timestep before present
SFV	external mode component of V
SFVB	external mode component of V for the timestep before present
SINE	sine of U,V point latitudes
SINEA	sine of U,V point latitudes in the Earth-oriented spherical coordinates
TINIT	initial values of tracers used only when starting a run from scratch
TNG	tangent of U,V point latitudes
ZDZ	vertical position of bottom of levels

ZDZZ	vertical position of center of levels
ZUN	time change of vertically averaged zonal forcing at north face
ZUS	time change of vertically averaged zonal forcing at south face
ZVN	time change of vertically averaged meridional forcing at north face
ZVS	time change of vertically averaged meridional forcing at south face

/SCALAR/

ACOR	program variable which is assigned the value of ACORF. If equal 0, treat the Coriolis term explicitly. If greater than 0, treat the Coriolis term implicitly with forward component weighted by ACOR
AH	program variable which is assigned the value of AHF; coefficient of horizontal mixing of T; = 1.Et7
AM	program variable which is assigned the value of AMF; coefficient of horizontal mixing of U,V
CRIT	program variable which is assigned the value of CRITF; criterion for convergence of relaxation; = 1.E8
DTSF	program variable which is assigned the value of DTSFF; length of the timestep on the stream function; = 180.0
DTTS	program variable which is assigned the value of DTTSF; length of the timestep on the tracer elements; = 1800.0
DTUV	program variable which is assigned the value of DTUVF; length of the timestep on U,V; = 180.0
FKPH	program variable which is assigned the value of FKPH; coefficient of vertical mixing of T; = 1.0
FKPM	program variable which is assigned the value of FKPMF; coefficient of vertical mixing of U,V
GRAV	acceleration due to gravity; = 980.6
OMEGA	rate of rotation of the coordinate system; = PI/43082
RADIAN	radian to degree conversion factor; = 59.29578
RADIUS	radius of the earth; = 6370.E5
SOR	program variable which is assigned the value of SORF; coefficient of over-relaxation; = 1.60
SWLDEG	latitude in degrees of the southern wall; = – 45.72

/TSTOP/

CHENG	assigned the Levitus climatological data of temperature and salinity

/WORKSP1/

FKMT	number of vertical levels of ocean at T points
FKMU	number of vertical levels of ocean at U,V points
FMM	FM at row J – 1
T	tracer variables, temperature and salinity
TB	tracer variables for the timestep before present
TDIF	diffusion computation array
TEMPA	initialized to 0
TEMPB	initialized to 0
U	zonal component of velocity
UB	zonal component of velocity for timestep before present
UUNDER	initialized to 0
V	meridional component of velocity
VB	meridional component of velocity for timestep before present

VUNDER initialized to 0

Parameters:
stored in file ocean.par

GRID	0.5715
GRID2	assigned to DYT and DXT, equal to GRID/INTVL2; = 0.28575
GRID3	assigned to DYT and DXT, equal to GRID/INTVL3; = 0.28575
IMT	total number of T grid boxes zonally; = 360
IMTKM	IMT * KM; = 5400
IMTM1	IMT – 1; = 359
IMTM2	IMT – 2; = 358
IMTP1	IMT + 1; = 361
IMU	total number of U,V grid boxes zonally; = 359
IMUM1	IMU – 1
IMUM2	IMU – 2
INT2	INTVL2 – 1; =1
INT3	INTVL3 – 1;=1
INTVL2	2
INTVL3	2
IRV	number of ocean basin grid cells for which river runoff is to be considered; = 88
JMT	total number of T grid boxes meridionally; = 360
JMTM1	JMT – 1, = 359
JMTM2	JMT – 2; = 358
JMTP1	JMT + 1; = 361
JSCAN	JMTM2
KM	total number of vertical levels in ocean basin; = 15
KMM1	KM – 1; = 14
KMP1	KM + 1; = 16
KMP2	KM + 2; = 17
LBC	number of arrays of slab data; = 2
LSEG	maximum number of sets of start and end indices for islands in ocean basin; = 10
MIDX	100
MIDXX	MIDX + INT2 + INT3 * (MIDX – (NSTLF1 + 1)); = 200
MIDY	80
MIDYY	MIDY + INT2 + INT3 * (MIDY – (NSTLW1 + 1)); = 160
NDICES	total number of start and end indices – 2 * LSEG * JMT + 4 * NISLE; = 7216
NGRDP	181
NGRDPX	NGRDP + INT2 + INT3 * NRTLF1 + INT2; = 361
NGRDPY	NGRDP + INT2 + INT3 * NUPLW1 + INT2; = 361
NIEVEN	2 * ((NISLE + 1)/2); = 4
NISLE	number of islands in the model basin; = 4
NKFLDS	number of two-dimensional fields needed on disk unit; 7 + (NDICES/NWDS)
NRTLF1	(NSTRT1–1) – (NSTLF1+1); = 178
NSLAB	number of words in one slab; IMT * ((NT + 2) * KM + LBC); = 22320
NSTLF1	0
NSTLW1	0
NSTRT1	180
NSTUP1	180
NSWICH	number of words of slab incidental data within each slab that are nonprognostic; NSWICH words must be switched into correct arrays after being read in on an even timestep; LBC * IMT; = 720

NT	number of tracer elements carried in the model (temperature and salinity); = 2
NTMIN2	NT + 1/NT
NUPLW1	(NSTUP1–1) – (NSTLW1+1); = 178
NWDS	IMT * JMT; = 129600
STLF1	0
STLW1	0
STRT1	180.0
STUP1	180.0

Local Data Files or Databases:

The ocean restart data are read from and written to file fort_<*yymmdd*>.21, where *yymmdd* is the year, month, and day of the run, using the Ocean Direct Access Manager (ODAM) software described in CSC Direct Access Manager (Sec. 3.4.6). Input restart data are read from logical unit 18 and output restart data are written to logical unit 34.

Logic Flow:

Initialize model data:

1. Define parameters through PARAMETER statement. This PARAMETER statement will be repeated in several CSUs.

2. Assign values to program constants. Set x, y, and z dimensions of the grid boxes in centimeters. Assign logical unit numbers. Set radius of the Earth, rate of rotation, acceleration due to gravity, and latitude in degrees of the southern wall of the model basin. Set PI and radian to degree conversion factor.

3. Read in file variables:

   a. Read data from the restart file of the ice model. This file contains the ice conditions of the current day including the ice-drift velocity, ice thickness, and ice concentration.

   b. Read ice growth rate in open water, total ice growth rate of the day, and heat above the freezing temperature.

   c. Read Earth-oriented latitudes of each gridpoint.

   d. Read run parameters including control parameters, eddy coefficients, island box corner point indices, initial values of tracers, timestep lengths for T,U,V, and stream function.

4. Compute auxiliary arrays based upon the specified spacing. Methods for computing these auxiliary arrays are straightforward using variable definitions given in the Data Element section of this CSU description.

   a. For Z direction, compute C2DZ, DZ2R, DZZ, ZDZ, DZZ2R, ZDZZ, EEH, FFH, EEM, and FFM. Auxiliary arrays are used for ease in enabling vectorization in other CSUs:
   $EEH_k = FKPH/(DZ_k * DZZ_k)$,
   $FFH_k = FKPH/(DZ_k * DZZ_{k+1})$,
   EEM and FFM are defined similarly with FKPM replacing FKPH.

   b. For y direction, compute PHI, PHIT, SUMDY, DYU, DYT, DYTR, DYT2R, DYT4R, DYUR, DYU2R, DYU4R, CST, CS, SINE, CSTR, CSR, and TNG.

   c. For x direction, compute DXU, DXTR, DXT2R, DXT4R, DXUR, DXU2R, and DXU4R

5. Open the disk datasets including slab data files, control data files, and two-dimensional horizontal field storage files.

Prepare to timestep:

At this point, program execution continues without consideration as to whether this is a model restart or a run from scratch.

1. Read disk data into memory for startup.
   a. Read in timestep counter, total elapsed time, surface area of the model basin, and total volume of the model basin from logical unit number KONTRL.
   b. Read start and end indices from logical unit number KFLDS and convert to integers by storing in ISZ and IEZ array.
   c. Compute permuting disk indicators and read in mass transport stream functions computed for current and previous timesteps. Also read in reciprocal depth array. These values are also stored in the file opened on logical unit number KFLDS.
2. Initialize array variables to 0 including elements of vertical mixing computation arrays, change in vorticity arrays, temporary storage arrays, and diffusion computation arrays.

Timestep the model:

Method: Loop to call CSU OSTEP to compute data for each timestep. Save restart data after every NWRITE calls to CSU OSTEP. After the specified number of timesteps have been taken, close disk units and stop execution.

3.4.2 CSC Ocean Timestep

This CSC controls the processing for each timestep. It prepares the variables needed to compute data for the timestep being processed and drives the main routines for computing tracer and velocity data.

3.4.2.1 CSU OSTEP

CSU OSTEP is the main routine for performing the functions required by CSC Ocean Timestep. It initializes various quantities and bootstraps and manages the I/O for the row-by-row computation of prognostic variables. It also performs analysis procedures on the progressing solution.

3.4.2.1.1 CSU OSTEP Design Specification/Constraints – There are no known constraints.

3.4.2.1.2 CSU OSTEP Design

Input Data Elements:

/FIELDS/

HR	reciprocal of total depth at U,V points
PB	mass transport stream function computed for timestep before present

/FULLWD/

EB	
ITT	timestep counter, incremented by 1 in this CSU

KAR	KAR(K) = K; for future vectorization
KFLDS	disk unit number of two-dimensional horizontal fields and start and end indices; = 12
KONTRL	disk unit number for timestep counter, etc.
LABS	disk unit numbers for slabs; = 13, 14, 15
NLAST	final timestep to compute on this run of the model
NMIX	number of timesteps between mixing timesteps
NNERGY	number of timesteps between printout of energy data
NTSI	number of timesteps between printout of timestep information
TTSEC	total elapsed time, incremented by the length of the timestep in this CSU
VOLUME	volume of the ocean basin

/ONEDIM/

C2DZ	DZ * 2
CS	cosine of U,V point latitudes
CSR	reciprocal of cosine of U,V point latitudes
CST	cosine of T point latitudes
DXT	zonal grid spacing across T boxes (between U,V points)
DXT4R	1.0/(DXT * 4.0)
DXU	zonal grid spacing across U,V boxes (between T points)
DXU2R	1.0/(DXU * 2.0)
DYU	meridional grid spacing across U,V boxes (between T points)
DYUR	1.0/DYU
DYU2R	1.0/(DYU * 2.0)
DZ	vertical grid spacing down U,V,T boxes (between W points); vertical laye thickness in centimeters
DZZ	vertical grid spacing across W boxes (between U,V,T points); assigned t DZZQ in this CSU
DZZ2R	1.0/(DZZ * 2.0); assigned to DZZ2RQ in this CSU
EEH	upper vertical mixing coefficient of T; assigned to EEHQ in this CSU
EEM	upper vertical mixing coefficient of U,V; assigned to EEMQ in this CSU
FFH	lower vertical mixing coefficient of T; assigned to FFHQ in this CSU
FFM	lower vertical mixing coefficient of U,V; assigned to FFMQ in this CSU
SINEA	sine of U,V point latitude in Earth-oriented spherical coordinates

/SCALAR/

AH	coefficient of horizontal mixing of T
DTSF	length of timestep on the mass transport stream function
DTTS	length of timestep on T
DTUV	length of timestep on U,V
OMEGA	rate of rotation of the coordinate system

/TSTOP/

WINDSX	zonal wind velocity
WINDSY	meridional wind velocity

/WORKSP1/

FKMT	number of vertical levels of ocean at T points for row J
FKMTP	number of levels at T points for row $J+1$; read in from slab data file
FKMUP	number of levels at U points for row $J+1$; read in from slab data file
TA	similar to T but after present
TB	tracer data computed for timestep before present
TBP	tracer data for the timestep before present in row $J+1$; set to TP on a mixing timestep
TDIF	diffusion computation array
TP	tracer data for row $J+1$; read from slab data files
UP	U component of horizontal velocity for row $J+1$; read from slab data files
V	meridional component of velocity
VP	V component of horizontal velocity for row $J+1$; read from slab data files

Output Data Elements:

/CURNTS/

T0	set equal to T for the last timestep processed
U0	set equal to U for the last timestep processed
V0	set equal to V for the last timestep processed

/FIELDS/

P	mass transport stream function output to disk
PB	mass transport stream function for the timestep before the present

/FULLWD/

BUOY	energy transfer through buoyancy effects
DTABS	volume average of absolute change of temperature
EKTOT	total kinetic energy normalized by volume
ENGEXT	accumulator of rate of change of kinetic energy of external mode; initialized in this CSU
ENGINT	accumulator of rate of change of kinetic energy of internal mode; initialized in this CSU
ITT	updated timestep counter
KMT	number of vertical levels of the ocean at T points
KMTP	array containing number of vertical levels of ocean at T points in row $J+1$
KMU	number of vertical levels of the ocean at U,V points
KMUP	array containing number of vertical levels of ocean at U,V points in row $J+1$
MIX	mixing timestep indicator set in this CSU to indicate whether the current timestep is a mixing timestep
MXP	set to 1 to indicate second pass of a Euler backward timestep
NDISK	permutes with NDISKA and NDISKB on logical units 13, 14, and 15 to access slabs at each timestep
NDISKA	see NDISK
NDISKB	see NDISK
NERGY	set to 1 on an energy printout timestep
PLICEX	energy change due to implicit effects on external mode
PLICIN	energy change due to implicit effects on internal mode
TTDTOT	initialized to 0
TTSEC	total elapsed time in seconds
TVAR	change of variance of tracers

/ONEDIM/

SFU	external mode component of U
SFUB	external mode component of U in timestep before present
SFV	external mode component of V
SFVB	external mode component of V in timestep before present
ZUS	initialized time change of vertically averaged zonal forcing at south face
ZVS	initialized time change of vertically averaged meridional forcing at south face

/RVR/

TRIVER	river temperature

/SCALAR/

C2DTSF	2 * DTSF; on a mixing timestep equal to DTSF
C2DTTS	2 * DTTS; on a mixing timestep equal to DTTS
C2DTUV	2 * DTUV; on a mixing timestep equal to DTUV

/WORKSP1/

BCON	array of slab incidental data on $N+1$ slab
C2DZQ	set equal to C2DZ for future vectorization
DXT4RQ	set equal to DXT4R for future vectorization
DXTQ	set equal to DXT for future vectorization
DXU2RQ	set equal to DXU2R for future vectorization
DXUQ	set equal to DXU for future vectorization
DZ2RQ	set equal to DZ2R for future vectorization
DZZ2RQ	set equal to DZZ2R for future vectorization
DZZQ	set equal to DZZ for future vectorization
EEHQ	set equal to EEH for future vectorization
EEMQ	set equal to EEM for future vectorization
FFHQ	set equal to FFH for future vectorization
FFMQ	set equal to FFM for future vectorization
FKMUP	number of vertical levels of the ocean at U,V points for row $J+1$
FM	masking array for T points
FMM	masking array for T points in row $J-1$
FMP	masking array for T points in row $J+1$
FVST	initialized advective coefficient for south face of T box
FVSU	advective coefficient for south face of U,V box
GM	masking array for U,V points
RHOS	initialized values of RHO for the row to the south of the present row
TA	tracer variables for timestep after present
TBM	tracer variables for timestep before present for row $J-1$
TBP	tracer variables for timestep before present for row $J+1$
TDIF	diffusion computation array
TM	tracer variables for row $J-1$
UBP	U component of horizontal velocity for timestep before present for row $J+$
UCLIN	array of internal mode component of U at row $J+1$
UP	U component of horizontal velocity for row $J+1$
UUNDER	positioned in common to be equivalent with UDIF with K index equal t $KM+1$
VBP	V component of horizontal velocity for timestep before present for row $J+$
VCLIN	array of internal mode component of V at row $J+1$
VUNDER	positioned in common to be equivalent with VDIF with K index equal to $KM+$

WSX	zonal component of surface wind stress
WSY	meridional component of surface wind stress
ZUSENG	vertical average of U forcing at south face
ZVSENG	vertical average of V forcing at south face

Parameters:
stored in file ocean.par
The ocean.par parameters are described in Sec. 3.4.1.1.2.

Logic Flow:
Initialize quantities needed for each timestep:

1. Increment timestep counter by 1 and the total elapsed time by timestep increment.

2. Update permuting disk I/O units. At any one time, three files are open for the storage and retrieval of slab and slab incidental data for the $N-1$ (timestep before present), N (present timestep), and $N+1$ (timestep after present) timesteps. For convenience, three permuting disk units are used. For details on the permutation method used by the I/O subsystem refer to Sec. 3.4.6.

3. Prepare timestep length variables C2DTTS, C2DTUV, and C2DTSF. If this is not a mixing timestep, set variables equal to **twice** the length of the timestep on T, U, and V, and the mass transport stream function. If this is a mixing timestep, set the same variables equal to the length of the timestep on T, U, and V, and the stream function. Also, current mass transport stream function should be saved to the previous mass transport stream function on a mixing timestep.

4. Call CSU STRESSUP to read NOGAPS geostrophic winds file and compute surface wind stress if this is the first timestep.

5. Establish over-dimensioned arrays for vectorization.

6. Call CSU STINIT to load coefficient arrays for subsequent calls to CSU STATE and CSU STATEC.

7. Reset U and V components of the vertical mixing computation arrays UUNDER and VUNDER to 0 for the $KM+1$ boundary.

8. Initialize arrays to contain volume average of absolute change of temperature (DTABS) and change of variance of tracers (TVAR).

9. If this is an energy printout timestep, set indicator flag and prepare necessary variables. Initialize arrays to accumulate rates of change of kinetic energy of internal and external modes (ENGINT, ENGEXT) and vertical average of V forcing at north and south faces of box (ZUSENG, ZVSENG). Also, initialize tracer arrays including arrays to contain integrals (TTDOT), and meridional and northward mass transport of tracers (TMT, TTN).

Bootstrap the row-by-row computation of prognostic variables:

This algorithm bootstraps the row-by-row processing by calculating data for row 1 for the timestep after present.

1. Call CSU OGET to retrieve row 2 tracer data from the disk for the timestep before present and the present timestep.

2. If this is an even timestep, switch slab incidental data into correct arrays. On even timesteps, slab incidental data is read into incorrect arrays because of the method used by the direct access subsystem. See description of I/O subsystem in Sec. 3.4.6.

3. Assign arrays containing maximum level indicators at U,V, and T points to integer arrays for use in equations.

4. If this is a mixing timestep, move T and U,V data from the timestep before present to the present timestep for row $J+1$.

5. Initialize arrays for first calls to CSU CLINIC and CSU TRACER including slab arrays, south face of T box advective coefficient arrays, and row J and $J-1$ land/water masking arrays for T points (FM,FMM). Set all array values to 0.

6. Construct masking array for row 2 T points by checking the contents of the array containing the number of vertical levels at T points. Assign 0 to land points; 1 to water points.

7. Initialize vorticity computation arrays at southern wall (ZUS, ZVS) to 0.

8. Save internal mode velocities for row 2 to arrays UCLIN and VCLIN and compute advective coefficient for south face of row 2 U,V boxes:

   $FVSU_{i,k} = (VP_{i,k} + V_{i,k}) * FX,$

   where

   $FX = DYU2R_2 * CSR_2 * CST_2 * 0.5.$

9. Compute external mode velocities for row 2 for the timestep before present and the present timestep from the mass transport stream function results read in CSU OCEAN in preparing to timestep:

   $SFU_i = ((P_{i+1,j+2} - P_{i,j+1}) + (P_{i,j+2} - P_{i+1,j+1})) * DYU2R_{j+1} * HR_{i,j+1},$

   $SFV_i = ((P_{i+1,j+2} - P_{i,j+1}) - (P_{i,j+2} - P_{i+1,j+1})) * DXU2R_i * HR_{i,j+1} * CSR_{j+1}.$

   On timestep before present, P is replaced by PB, and SFU and SFV are replaced by $SFUB$ and $SFVB$.

10. Add external mode to internal mode for the ocean points in row 2 for present timestep and timestep before present.

11. Accumulate kinetic energy from row 2 every $NTSI$ timesteps:

    $EKTOT = EKTOT + (UP_{i,k}^2 + VP_{i,k}^2) * 0.5 * CS_{j+1} * DYU_{j+1} * DZ_k * DXU_i.$

12. Call CSU STATE to compute density of row 2.

Perform row-by-row computation of prognostic variables:

Loop for rows 2 through $JMT-1$:

1. Move tracer data for present timestep and timestep before present down one row. Move array elements for row J data to array elements for row $J-1$. Move row $J+1$ data to row J.

2. Except when processing the last row, read the rest of the $J+1$ slab for the timestep before present and the present timestep.

3. Except when processing row 2 (the first row processed in this algorithm), write newly computed data from the previous row, row $J-1$. This data was computed in the previous execution of this row-by-row processing loop.

4. If processing an even timestep, switch maximum level indicators, read in step 2 above into correct slab. Because of the permuting disk I/O, the maximum level indicators for the U,V points are in the *FKMTP* array and the maximum level indicators for the T points are in the *FKMUP* array. The arrays are switched on even timesteps only as is explained in Sec. 3.4.6.

5. Shift maximum level indicators for T and U,V points from row $J+1$ to row J and set $J+1$ floating point values to integer. Maximum level indicators should be in arrays *KMT* and *KMU* for row J and *KMTP* and *KMUP* for row $J+1$.

6. If on a mixing timestep, set tau – 1 slab data equal to tau level slab data. Slab data includes tracer data and U and V components of horizontal velocity.

7. Shift masks down one row and compute new masks. The masking arrays are *FM* for row J, *FMM* for row $J-1$, *FMP* for row $J+1$. Set new values for row J into *FMP* based on contents of *KMTP* arrays. Also set *GM* masking array based on contents of *KMU*.

8. Call CSU CLINIC to update the internal mode and vorticity driving function for row J.

9. Call CSU TRACER to update the tracer quantities for row J.

10. Print the progressing solution at every fourth row on energy timestep. Call CSU MATRIX to print arrays of temperature and salinity tracers and U,V, and W components of velocity.

11. If not processing the northernmost row, compute the northward transport of each tracer quantity, as well as the zonally integrated meridional mass transport.

    a. For temperature and salinity tracers, move data computed for the north of the previous row to the south of the current row.

    b. Sum all tracer data for row 2, masking out land points with the *FM* array so that land grid cells do not contribute to the sum. Average tracer data by dividing by the total ocean area of row 2:

    $TBRS_{k,m} = TBRS_{k,m} + T_{i,k,m} * FM_{i,k} * DXT_i,$

    $TBRS_{k,m} = TBRS_{k,m}/TOTDX,$

    where *TOTDX* is the sum of the zonal grid cells for all ocean gridpoints in row 2. Row 2 land points are not included in the sum.

    c. Compute *TBRN* similarly using masking array and data computed for row $J+1$ in timestep before present.

    $TBRN_{k,m} = TBRN_{k,m} + TP_{i,k,m} * FMP_{i,k} * DXT_i,$

    $TBRN_{k,m} = TBRN_{k,m}/TOTDX.$

    d. Sum the horizontal velocity zonally.

    $VBR_k = VBR_k + V_{i,k} * DXU_i * CS_j.$

    e. Compute the meridional mass transport by successively adding *VBR* contributions at each depth level.

For depth level 1:

$TMT_{j,1} = VBR_1 * DZ_1$.

For all other depth levels:

$TMT_{j,k} = TMT_{j,k-1} + VBR_k * DZ_k$.

f. Compute *TTN*, the northward transport of the tracers:

$TTN_{1,j,m} = TTN_{1,j,m} + VBR_k * (TBRN_{k,m} + TBRS_{k,m}) * 0.5 * DZ_k$

g. Compute zonal/vertical averages of tracers and horizontal velocity:

$VBRZ = VBRZ + (V_{i,k} * DXU_i + V_{i-1,k} * DXU_{i-1}) * DZ_k$,

$VBRZ = VBRZ/TOTDZ$,

where *TOTDZ* is the total vertical grid spacing,

$TBRZ = TBRZ + T_{i,k,m} + TP_{i,k,m} * DZ_k$.

h. If processing the last timestep, write data to the file connected to logical unit 10. This is the file containing the ocean conditions of the current day.

12. Move slab incidental data into the correct slab for writing using the temporary array *BCON*. On an even timestep, set *BCON* values equal to *FKMT* values. Otherwise, set *BCON* values equal to *FKMU* values. These data are written out at the beginning of the next execution of this loop (see step 3 above of this algorithm).

End loop for row-by-row computation.

Write out computed data:

1. Print one line of timestep information every *NTSI* timesteps, including timestep number, total kinetic energy, volume average of absolute change in temperature and salinity, and number of scans to convergence in CSU CRELAX.

2. Complete and print the on-line integrals on energy timesteps.

   a. Normalize previously computed integrals by dividing by volume. This includes total change of kinetic energy on internal and external modes, tracer integrals, and total change of variance on tracers.

   b. Compute residuals for energy change due to implicit effects on internal and external modes and tracer integrals.

3. Convert heat transport to Petawatts using conversion factor 4.186E – 15 and convert salt transport to 10^{10} cm^3/s using conversion factor 1.E – 10.

4. Initiate write-out of newly computed data from the final row.

Prepare for the next timestep:

1. Call CSU CRELAX to solve for the new mass transport stream function.

2. If this is the end of the first pass of a Euler backward timestep, set the input disk units so that the proper levels are fetched on the next pass. The output for the second pass will be placed on the *NDISKA* unit. Return to the top of CSU OSTEP to do the second pass if this is the first pass of a Euler backward timestep.

3. For purposes of recovering from the disk after an abnormal stop, bring otherwise inactive disk units up-to-date by writing the new mass transport stream function and the total number of timesteps that were completed.

4. If this is an energy timestep, print the new stream function using CSU MATRIX.

3.4.2.2 CSC Compute Wind/Ice Stress

CSC Compute Wind/Ice Stress requires the computation of the wind/ice stress using a wind drag coefficient which varies as a function of ice thickness.

3.4.2.2.1 CSU STRESSUP

CSU STRESSUP meets all of the requirements specified for the CSC Compute Wind/Ice Stress.

3.4.2.2.1.1 CSU STRESSUP Design Specification/Constraints. Compute wind/ice stress using a wind drag coefficient which varies as a function of ice thickness according to the law of the wall.

3.4.2.2.1.2 CSU STRESSUP Design

Input Data Elements:

/COX2/

FW1	heat above the freezing temperature
GICE	ice growth rate computed by the ice model
SHICE	growth rate of ice thickness

/CURNTS/

T0	set equal to T for the last timestep processed
U0	set equal to U for the last timestep processed
V0	set equal to V for the last timestep processed

/FULLWD/

EKTOT	total kinetic energy normalized by volume
ENGEXT	accumulators of rates of change of kinetic energy of external mode
ENGINT	accumulators of rates of change of kinetic energy of internal mode
ITT	timestep counter; total number of timesteps completed
KAR	KAR(K) = K; used to enable vectorization
KMU	number of vertical levels of ocean at U,V points
KMUP	number of vertical levels of ocean at U,V points for row J + 1
MXP	if = 1, second pass of Euler backward timestep
NERGY	if = 1, indicates energy printout timestep
NTSI	number of timesteps between prints of a single line of timestep information

/RFOR2/

GAIRX	x component of the geostrophic wind
GAIRY	y component of the geostrophic wind

/RSTRT/

AREA1	fraction of grid cell covered with ice
HEFF	mean ice thickness per grid cell

UICE	x component of ice drift
VICE	y component of ice drift

/TSTEP/

IDTG	date-time group in the form YYMMDDHH
ITSTEP	number of timesteps for run
MDY	day calculated from inputted data
MHR	hour calculated from inputted data
MM	month calculated from inputted data
MYR	year calculated from inputted data

/TSTOP/

WINDSX	zonal wind velocity
WINDSY	meridional wind velocity

Output Data Elements:

/TSTOP/

WINDSX	zonal wind velocity
WINDSY	meridional wind velocity

Parameters:

stored in file ocean.par

The ocean.par parameters are described in Sec. 3.4.1.1.2.

Local Data Elements:

CDWI	drag coefficient between water and ice as a function of ice thickness according to the law of the wall
CONSTK	the von Karman constant, 0.41
CONSTZ	minimum sea ice roughness, 0.01 m
HMIN	minimum ice thickness
I	index counter
J	index counter
TX	temporary array used in computing wind stress
TY	temporary array used in computing wind stress

Logic Flow:

1. For all surface grid cells except the last row and column, compute wind/ice stress using compute wind drag coefficient, *CDWI*.

   Compute wind drag coefficient:

   $CDWI = 1.0/(DLOG(HMIN/CONSTZ)/CONSTK) * * 2,$

   where

   HMIN = maximum of the ice thickness for grid cell *I,J* and 0.015,
   CONSTZ = 0.01, and
   CONSTK = 0.41.

Compute:

$$TX_{i,j} = WINDSX_{i,j} * |WINDSX_{i,j}| * 0.0013 * 0.0008 * 10000.0 * (1.0 - AICE_{i,j,2}) + (UICE_{i,j,2} * 100.0 - U1_{i,j}) * |UICE_{i,j,2} * 100.0 - U1_{i,j}| * CDWI * AICE_{i,j,2}$$

$$TY_{i,j} = WINDSY_{i,j} * |WINDSY_{i,j}| * .0013 * .0008 * 10000. * (1.0 - AICE_{i,j,2}) + (VICEi_{j,2} * 100. - V1_{i,j}) * |VICE_{i,j,2} * 100. - V1_{i,j}| * CDWI * AICE_{i,j,2}$$

2. Set *WINDSX* and *WINDSY* for all surface grid cells. Last row and column values will be 0. $WINDSX_{i,j} = TX_{i,j}/10.0$ and $WINDSY_{i,j} = TY_{i,j}/10.0$.

3.4.2.3 CSC Load Normalization Constants

This sublevel CSC requires a CSU to load normalization constants into arrays for use in computing densities.

3.4.2.3.1 CSU STINIT – CSU STINIT loads normalization constants into arrays for subsequent calls to CSU STATE and CSU STATEC.

3.4.2.3.1.1 CSU STINIT design specification/constraints. Overdimension arrays are provided to permit future vectorization.

3.4.2.3.1.2 CSU STINIT design.

Output Data Elements:

/WORKSP/

CQ	coefficients of normalized temperatures and salinities used in CSU STATE
TOQ	normalizing temperatures used in CSU STATE
SOQ	normalizing salinities used in CSU STATE
CIQ	coefficients of normalized temperatures and salinities used in CSU STATEC
TOIQ	normalizing temperatures used in CSU STATEC
SOIQ	normalizing salinities used in CSU STATEC

Local Data Elements:

C	coefficients of equation of state assigned in a DATA statement
SO	normalizing salinities
TO	normalizing temperatures

Parameters:

stored in file ocean.par

The ocean.par parameters are described in Sec. 3.4.1.1.2.

Logic Flow:

Load coefficients of equation of state into overdimensioned array for use in CSU STATE.

Load normalizing temperatures and salinities into overdimensioned arrays for use in CSU STATE.

Load coefficients of equation of state for use in CSU STATEC. Determine the reference level indicator so that the coefficients for the two levels being compared are equal. Two passes are required to set two sets of coefficients. In the first pass, values for the first set of elements are assigned, where levels 1 and 2 are assigned level 2 coefficients, levels 3 and 4 are assigned level 4 coefficients, and so on. In the second pass, values for the second set of elements are

assigned, where levels 1 and 2 are assigned level 1 coefficients, levels 3 and 4 are assigned level 3 coefficients, and so on.

Load normalizing temperatures and salinities into overdimensioned arrays for use in CSU STATEC using the method described above.

3.4.2.4 CSC Compute Densities

CSC Compute Densities requires the computation of normalized densities.

3.4.2.4.1 CSU STATE and CSU STATEC – CSU STATE and CSU SINIT are used to compute normalized densities using a third-order fit to the Knudsen formula. CSU STATE is called by CSU CLINIC and by CSU TIMESTEP in the bootstrap procedure. CSU STATEC is called by CSU TRACER.

3.4.2.4.1.1 CSU STATE and CSU STATEC design specification/constraints. This subroutine contains two entry points for computing density.

3.4.2.4.1.2 CSU STATE and CSU STATEC design.

Input/Output Data Elements:

RHO	the returned row of normalized densities
TX	the input row of temperatures
SQ	one row of workspace provided by the calling routine
TQ	one row of workspace provided by the calling routine
SX	the input row of salinities in units (ppts–35)/1000
IND	for CSU STATEC only: if IND = 1, compare levels 1 to 2, 3 to 4, etc.; if IND = 2, compare levels 2 to 3, 4 to 5, etc. in either case, use coefficients for the lower of the two levels

/WORKSP/

SOQ	normalizing salinities
TOQ	normalizing temperatures

Parameters:
stored in file ocean.par

The ocean.par parameters are described in Sec. 3.4.1.1.2.

Logic Flow:

Compute normalized densities by using a third-order polynomial fit to the Knudsen formula

1. Subtract normalizing constants from input temperature and salinity.

2. Compute polynomial approximation of Knudsen density:

$$RHO_{i,k} = (CQ_{i,k,1} + (CQ_{i,k,4} + CQ_{i,k,7} * SQ_{i,k}) * SQ_{i,k} + (CQ_{i,k,3} + CQ_{i,k,8} * SQ_{i,k} + CQ_{i,k,6} * TQ_{i,k}) * TQ_{i,k}) + (CQ_{i,k,2} + CQ_{i,k,5} + CQ_{i,k,9} * SQ_{i,k}) * SQ_{i,k}^2$$

For entry *STATEC*, array *CIQ* replaces array *CQ* and the index of the fourth dimension of *CIQ* is equal to *IND*.

3.4.2.5 CSC Print Matrix

CSC Print Matrix requires a subroutine to print elements of a two-dimensional array.

3.4.2.5.1 CSU MATRIX – CSU MATRIX is called by CSU OSTEP to print elements of a two-dimensional matrix on specified timesteps.

3.4.2.5.1.1 CSU MATRIX design specification/constraints.

3.4.2.5.1.2 CSU MATRIX design.

Input Data Elements:

ARRAY	the array to be printed
IRDIM	the first dimension of the array
ISTRT	the first element of the first dimension to be printed
IM	the last element of the first dimension to be printed
JM	the last element of the second dimension to be printed – the rows are printed in descending order (if $JM = 0$, KK is used)
KK	the last element of the second dimension to be printed – the rows are printed in ascending order (if $KM = 0$, JM is used)
SCALE	a scaling factor by which array is divided before printing. If scale = 0, no scaling is done and 10 columns are printed across in E format. If scale > 0, scaling is performed and 20 columns are printed across in F format.

Local Data Elements:

IDIF	number of elements to be printed across a line; the difference between IE and IS plus 1
IE	ending element number when printing a line of values
IS	starting element number when printing a line of values
JORK	do loop index for second dimension
JMORKM	ending element of second dimension, equals $JM + KK$
L	the row number printed on a line
NUM	array for temporary storage, unscaled ARRAY elements printed across a line
PLINE	array for temporary storage, ARRAY elements multiplied by SCALER
SCALER	reciprocal of scale factor, used to multiple array elements

Parameters:
stored in file ocean.par

The ocean.par parameters are described in Sec. 3.4.1.1.2.

Logic Flow:

1. If SCALE = 0, print array elements in groups of 10. Print column numbers across the top. Print row number followed by 10 array elements using E13.5 format. Repeat until all desired values are printed.

2. Otherwise, print array elements in groups of 20. Print column numbers across the top. Print row number followed by 20 array elements using F6.2 format. Repeat until all desired values are printed.

3.4.3 CSC Compute Internal Mode and Vorticity Driving Function

This CSC requires computation of the internal mode component of the U and V velocities and the vorticity driving function for use in determining the external mode.

3.4.3.1 CSU CLINIC

CSU CLINIC meets the requirements of CSC Compute Internal Mode and Vorticity Driving Function.

3.4.3.1.1 CSU CLINIC Design Specification/Constraints – There are no known constraints.

3.4.3.1.2 CSU CLINIC Design

Input Data Elements:

/FULLWD/

EKTOT	total kinetic energy normalized by volume
ENGEXT	accumulators of rates of change of kinetic energy of external mode
ENGINT	accumulators of rates of change of kinetic energy of internal mode
ITT	timestep counter; total number of timesteps completed
KAR	KAR(K) = K; used to enable vectorization
KMU	number of vertical levels of ocean at U,V points
KMUP	number of vertical levels of ocean at U,V points for row $J+1$
MXP	if = 1, second pass of Euler backward timestep
NERGY	if = 1, indicates energy printout timestep
NTSI	number of timesteps between prints of a single line of timestep information

/ONEDIM/

CS	cosines of U,V point latitudes
CSR	reciprocal of the cosines of the U,V point latitudes
CST	cosines of T point latitudes
CSTR	reciprocal of the cosines of T point latitudes
DXT	zonal grid spacing across T boxes
DXT2R	1.0/(DXT * 2.0)
DXT4R	1.0/(DXT * 4.0)
DXTR	1.0/DXT
DXU	zonal grid spacing across U,V boxes
DXU2R	1.0/(DXU * 2.0)
DXUR	1.0/DXU
DYT	meridional grid spacing across T boxes
DYT2R	1.0/(DYT * 2.0)
DYTR	1.0/DYT
DYU	meridional grid spacing across U,V boxes
DYU2R	1.0/(DYU * 2.0)
DYU4R	1.0/(DYU * 4.0)
DYUR	1.0/DYU
DZ	vertical grid spacing down U,V,T boxes; vertical layer thickness in centimeter
DZZ	vertical grid spacing down W boxes
EEM	upper vertical mixing coefficient of U,V
FFM	lower vertical mixing coefficient of U,V
SINE	sine of U,V point latitude
SINEA	sine of U,V point latitude in Earth-oriented coordinates; used to compu Coriolis force
TNG	tangent of U,V point latitude
ZUS	time change of vertically averaged zonal forcing at south face
ZVS	time change of vertically averaged meridional forcing at south face

/SCALAR/

ACOR	if >0, treat the Coriolis term implicitly with forward component weighted by ACOR, past component by 1 – ACOR
AM	coefficient of horizontal mixing of U,V
C2DTSF	DTSF * 2.0
C2DTUV	DTUV * 2.0
FKPM	coefficient of vertical mixing of U,V
GRAV	acceleration due to gravity; = 980.6 cm/s^2
OMEGA	rate of rotation of the coordinate system
RADIUS	radius of the Earth

/WORKSP1/

C2DZQ	DZ * 2.0; for future vectorization
DXT4RQ	DXT4R (1.0/(DXT * 4.0)); for future vectorization
DXU2RQ	DXU2R (1.0/(DXU * 2.0)); for future vectorization
DXUQ	DXU; for future vectorization
DZZQ	DZZ; for future vectorization
EEMQ	EEM ; for future vectorization
FFMQ	FFM; for future vectorization
FMM	lower vertical mixing coefficient of U,V
FVSU	advective coefficient for south face of U,V box
GM	masking array for U,V points
RHON	density with a space and time invariant subtracted, for the row to the north of the current row
RHOS	density with a space and time invariant subtracted, for the row to the south of the current row
TDIF	diffusion computation array
TP	tracer variables computed for row $J + 1$
U	zonal component of horizontal velocity
UB	zonal component of horizontal velocity for the timestep before the present timestep
UBM	UB for row $J - 1$
UBP	UB for row $J + 1$
UCLIN	array of internal mode component of U at row $J + 1$
UM	zonal component of horizontal velocity for row $J - 1$
UP	U for row $J + 1$
V	meridional component of horizontal velocity
VB	meridional component of horizontal velocity for the timestep before the present timestep
VCLIN	array of internal mode component of V at row $J + 1$
VM	V for row $J - 1$
VP	V for row $J + 1$
WSX	zonal component of surface wind stress
WSY	meridional component of surface wind stress
ZUSENG	vertical average of U forcing at south face
ZVSENG	vertical average of V forcing at south face

Output Data Elements:

/FULLWD/

EKTOT	total kinetic energy normalized by volume
ENGEXT	accumulators of rates of change of kinetic energy on external mode
ENGINT	accumulators of rates of change of kinetic energy on internal mode

/ONEDIM/

SFU	external mode component of U
SFUB	external mode component of U for timestep before present
SFV	external mode component of V
SFVB	external mode component of V for timestep before present
ZUN	time change of vertically averaged zonal forcing at north face
ZUS	time change of vertically averaged zonal forcing at south face
ZVN	time change of vertically averaged meridional forcing at north face

/WORKSP1/

FUW	zonal component of advective coefficient for west face of U,V box
FVN	meridional component of advective coefficient for north face of U,V box
FVSU	advective coefficient for south face of U,V box
TDIF	diffusion computation array
TEMPA	utility array used as temporary storage
TEMPB	utility array used as temporary storage
TP	similar to T but in row $J+1$
UA	zonal component of velocity for timestep after present

Parameters:

stored in file ocean.par

The ocean.par parameters are described in Sec. 3.4.1.1.2.

Logic Flow:

Prepare arrays for the computation of internal modes:

Several arrays must be calculated to prepare for the computation of the internal modes o horizontal velocity. These include:

- the advective coefficients for the west (*FUW*) and north (*FVN*) faces of the U,V boxes,
- the zonal and meridional components of external mode,
- horizontal velocities for row $J+1$ for tau and tau $-$ 1 time levels,
- the density for row $J+1$,
- the vertical velocity for U,V columns,
- hydrostatic pressure gradients, and
- boundary conditions for the computation of the vertical diffusion of momentum.

1. Find advective coefficients for west and north faces of U,V box using data computed from th timestep before present. These variables were read in CSU OSTEP:

   a. Calculate external mode component of U (*SFU*) at west face of U,V box and external mod component of V (*SFV*) at north face of U,V box. Use mass transport stream function rea in CSU OSTEP:

   $$SFU_i = -\ (P_{i,j+1} - P_{i,j}) * DYUR_j * HR_{min},$$

   where HR_{min} is the minimum of $HR_{i-1,j}$ and $HR_{i,j}$, and

   $$SFV_i = (P_{i+1,j+1} - P_{i,j+1}) * DXUR_i * HR_{min} * CSTR_{j+1},$$

   where HR_{min} is the minimum of $HR_{i,j+1}$ and $HR_{i,j}$.

b. Calculate internal mode component of U at west face of U,V box and internal mode component of V at north face of U,V box:

$FUW_{i,k} = (UCLIN_{i,k} + UCLIN_{i-1,k}) * 0.5$

$FVN_{i,k} = (VP_{i,k} + VCLIN_{i,k}) * 0.5.$

c. Add external modes to internal modes and add grid weight factor to obtain advective coefficients where

$CSR_j = 1/\cos(\text{latitude}_j)$

is the grid weight factor for the west face of U,V box and

$FX = DYU2R_j * CSR_j * CST_{j+1}$

is the grid weight factor for the north face of U,V box.

2. Compute external mode velocities for row $J + 1$ for present timestep and timestep before present. External mode velocities are computed from the mass transport stream function, P:

$SFU = -\ ((P_{i+1,j+2} - P_{i,j+1}) + (P_{i,j+2} - P_{i+1,j+1})) * DYU2R_{j+1} * HR_{i,j+1}$

$SFV = ((P_{i+1,j+2} - P_{i,j+1}) - (Pi,j+2 - P_{i+1,j+1})) * DXU2R_i * HR_{i,j+1} * CSR_{j+1}.$

On timestep before present, PB replaces P and results are stored in $SFUB$ and $SFVB$.

3. Add external mode to internal mode for all ocean points in row $J + 1$ for present timestep and timestep before present.

4. Accumulate kinetic energy from row $J + 1$ every NTSI timesteps.

Compute contribution to total kinetic energy:

$UENG_{i,k} = (FX * (UP_{i,}k^2 + VP_{i,k}^2)) * C2DZQ_{i,k} * DXUQ_{i,k},$

where $FX = 0.25 * CS_{j+1} * DYU_{j+1}.$

Add $UENG_{i,k}$ to total kinetic energy stored in $EKTOT$.

5. Call CSU STATE to compute density of row $J + 1$. Store computed results in RHON.

6. Compute vertical velocity in U,V columns.

a. Set vertical velocity at the surface to 0 (rigid-lid), and set vertical velocity at maximum level to 0. The rigid-lid assumption of 0 vertical motion at the surface filters out external gravity waves that would otherwise limit the timestep of the numerical integration.

b. Compute change of vertical velocity W between levels

$W_{i,k+1} = C2DZQ_{i,k} * ((FUW_{i+1,k} - FUW_{i,k}) * DXU2RQ_{i,k} + FVN_{i,k} - FVSU_{i,k}).$

c. Integrate downward from the surface by adding successive vertical velocities:

$W_{i,k+1} = W_{i,k} + W_{i,k+1}.$

7. Compute hydrostatic pressure gradient.

   a. Compute it at the first level:

   $UDIF_{i,1} = RHON_{i+1,1} - RHOS_{i,1}$

   $VDIF_{i,1} = RHON_{i,1} - RHOS_{i+1,1}$

   $DPDX_{i,1} = ((UDIF_{i,1} - VDIF_{i,1}) * FXA) * DXU2R_i$

   $DPDY_{i,1} = (UDIF_{i,1} + VDIF_{i,1}) * FXB,$

   where $FXA = GRAV * DZZ_1 * CSR_j$ and

   $FXB = GRAV * DZZ_1 * DYU2R_j.$

   b. Compute the change in pressure gradient between levels:

   $DPDX_{i,k} = RHON_{i,k-1} + RHON_{i,k}$

   $DPDY_{i,k} = RHOS_{i,k-1} + RHOS_{i,k}$

   $UDIF_{i,k} = DPDX_{i+1,k} - DPDY_{i,k}$

   $VDIF_{i,k} = DPDX_{i,k} - DPDY_{i+1,k}$

   $DPDX_{i,k} = (FXA * (UDIF_{i,k} - VDIF_{i,k})) * DZZQ_{i,k} * DXU2RQ_{i,k}$

   $DPDY_{i,k} = (FXB * (UDIF_{i,k} + VDIF_{i,k})) * DZZQ_{i,k}.$

   c. Integrate downward from the first level:

   $DPDX_{i,k+1} = DPDX_{i,k} + DPDX_{i,k+1}$

   $DPDY_{i,k+1} = DPDY_{i,k} + DPDY_{i,k+1}.$

8. Set the boundary conditions for the computation of vertical diffusion of momentum.

   a. Transfer interior points into diffusion computation arrays. Store *UB* and *VB* elements int*UDIF* and *VDIF* arrays.

   b. Set surface (K=0) elements of diffusion computation arrays to reflect wind stress:

   $UOVER_i = UB_{i,1} + WSX_i * FX,$

   $VOVER_i = VB_{i,1} + WSY_i * FX,$

   where $FX = DZZ_1/FKPM.$

   c. Set first land level in each column to reflect bottom condition. Assume a 10° turning angl at the bottom boundary:

   $UDIF_{i,kz+1} = UB_{i,kz} - FXB * (UB_{i,kz} * 0.98481 - VB_{i,kz} * 0.17365)$

   $VDIF_{i,kz+1} = VB_{i,kz} - FXB * (UB_{i,kz} * 0.17365 + VB_{i,kz} * 0.98481).$

Begin computation of the internal modes.

Method: First, the total advection of momentum is calculated from the flux through the we face of the U,V box, the meridional and zonal flux divergence, the flux through the top of the U, box, and the vertical flux divergence. Next, horizontal diffusion of momentum is evaluated f tau – 1 timestep and vertical diffusion of momentum is determined from gradients at top of U,V bo and from upper and lower vertical mixing coefficients of U and V. Coriolis force is computed tau timestep for explicit treatment and at tau – 1 timestep for implicit treatment. Adding the

elements to the computed hydrostatic pressure gradient yields the time rate of change of velocity. New velocities are computed by multiplying the time rate of change by twice the length of the timestep on U,V and adding this product to the U and V components of velocity computed for the timestep before present. These velocities are corrected by finding the incorrect vertical means and subtracting them out.

1. Compute total advection of momentum.

   a. Compute flux through west face of U,V box:

   $$TEMPA_{i,k} = FUW_{i,k} * (U_{i-1,k} + U_{i,k})$$
   $$TEMPB_{i,k} = FUW_{i,k} * (V_{i-1,k} + V_{i,k})$$

   b. Compute zonal flux divergence:

   $$UA_{i,k} = (TEMPA_{i,k} - TEMPA_{i+1,k} * DXU2RQ_{i,k}$$
   $$VA_{i,k} = (TEMPB_{i,k} - TEMPB_{i+1,k}) * DXU2RQ_{i,k}$$

   c. Add in meridional flux divergence:

   $$UA_{i,k} = UA_{i,k} - FVN_{i,k} * (UP_{i,k} + U_{i,k}) + FVSU_{i,k} * (U_{i,k} + UM_{i,k})$$
   $$VA_{i,k} = VA_{i,k} - FVN_{i,k} * (VP_{i,k} + V_{i,k}) + FVSU_{i,k} * (V_{i,k} + VM_{i,k})$$

   d. Compute flux through top of U,V box:

   $$TEMPA_{i,k} = W_{i,k} * (U_{i,k-1} + U_{i,k})$$
   $$TEMPB_{i,k} = W_{i,k} * (V_{i,k-1} + V_{i,k})$$

   e. Add in vertical flux divergence:

   $$UA_{i,k} = UA_{i,k} + TEMPA_{i,k+1} - TEMPA_{i,k} * DZ2RQ_{i,k}$$
   $$VA_{i,k} = VA_{i,k} + TEMPB_{i,k+1} - TEMPB_{i,k} * DZ2RQ_{i,k}.$$

2. Add in horizontal diffusion of momentum (evaluated at tau – 1 timestep).

   a. Compute coefficients dependent only on latitude:

   $$BBUJ = 8.0 * AM * CSR_j^2$$
   $$CCUJ = AM * CST_{j+1} * DYTR_{j+1} * DYUR_j * CSR_j$$
   $$DDUJ = AM * CST_j * DYTR_j * DYUR_j * CSR_j$$
   $$GGUJ = AM * (1.0 - TNG_j^2)/RADIUS^2$$
   $$HHUJ = 2.0 * AM * SINE_j/(RADIUS * CS_j^2)$$

   b. Compute gradients at west face of U,V box:

   $$TEMPA_{i,k} = DXT4RQ_{i,k} * (UB_{i,k} - UB_{i-1,k})$$
   $$TEMPB_{i,k} = DXT4RQ_{i,k} * (VB_{i,k} - VB_{i-1,k})$$

   c. Add in final contribution from horizontal diffusion of momentum:

   $$UA_{i,k} = UA_{i,k} + BBUJ * (DXU2RQ_{i,k} * (TEMPA_{i+1,k} - TEMPA_{i,k})) + CCUJ * (UBP_{i,k} - UB_{i,k}) + DDUJ * (UBM_{i,k} - UB_{i,k}) + GGUJ * UB_{i,k} - HHUJ * DXU2RQ_{i,k} * (VB_{i+1,k} - VB_{i-1,k})$$

$$VA_{i,k} = VA_{i,k} + BBUJ * (DXU2RQ_{i,k} * (TEMPB_{i+1,k} - TEMPB_{i,k}) + CCUJ * (VBP_{i,k} - VB_{i,k}) + DDUJ * (VBM_{i,k} - VB_{i,k}) + GGUJ * VB_{i,k} + HHUJ * DXU2RQ_{i,k} * (UB_{i+1,k} - UB_{i-1,k}).$$

3. Add in vertical diffusion of momentum.

   a. Compute gradients at top of U,V box:

   $$TEMPA_{i,k} = UDIF_{i,k-1} - UDIF_{i,k}$$

   $$TEMPB_{i,k} = VDIF_{i,k-1} - VDIF_{i,k}.$$

   b. Add in final contribution from vertical diffusion of momentum:

   $$UA_{i,k} = UA_{i,k} + EEMQ_{i,k} * TEMPA_{i,k} - FFMQ_{i,k} * TEMPA_{i,k+1}$$

   $$VA_{i,k} = VA_{i,k} + EEMQ_{i,k} * TEMPB_{i,k} - FFMQ_{i,k} * TEMPB_{i,k+1}.$$

4. Add in Coriolis force. Evaluate at tau timestep for explicit treatment and at tau – 1 timestep fo implicit treatment with remainder of term to be added later.

   Explicit treatment of Coriolis force:

   $$UA_{i,k} = UA_{i,k} + FX * V_{i,k}$$

   $$VA_{i,k} = VA_{i,k} - FX * U_{i,k}.$$

   Implicit treatment of Coriolis force:

   $$UA_{i,k} = UA_{i,k} + FX * VB_{i,k}$$

   $$VA_{i,k} = VA_{i,k} - FX * UB_{i,k}.$$

   For both explicit and implicit treatment of Coriolis force, $FX = 2.0 * OMEGA * SINEA_{i,j}$.

5. Add in pressure term, masking out land with the GM masking array.

   $$UA_{i,k} = GM_{i,k} * (UA_{i,k} - DPDX_{i,k})$$

   $$VA_{i,k} = GM_{i,k} * (VA_{i,k} - DPDY_{i,k}).$$

6. Form time change of vertically averaged forcing.

   a. Integrate time change vertically:

   $$ZUN_i = ZUN_i + UA_{i,k} * FX$$

   $$ZVN_i = ZVN_i + VA_{i,k} * FX,$$

   where $FX = C2DTSF * DZ_k$.

   b. Form average by multiplying by reciprocal of depth (dividing by depth):

   $$ZUN_i = ZUN_i * HR_{i,j}$$

   $$ZVN_i = ZVN_i * HR_{i,j}.$$

7. Do analysis of internal mode forcing on energy timestep. Form vertical average for use lat in external mode analysis.

a. Compute change in kinetic energy due to pressure term:

$UENG_{i,k} = GM_{i,k} * - DPDX_{i,k}$,

$VENG_{i,k} = GM_{i,k} * - DPDY_{i,k}$,

$ENGINT_6 = ENGINT_6 + (USAV_{i,k} * UENG_{i,k} + VSAV_{i,k} * VENG_{i,k}) * FX * DXU_i * DZ_k$,

where $FX = CS_j * DYU_j$,

$ZUNENG_{i,6} = ZUNENG_{i,6} + UENG_{i,k} * DZ_k * HR_{i,j}$,

$ZVNENG_{i,6} = ZVNENG_{i,6} + VENG_{i,k} * DZ_k * HR_{i,j}$.

b. Compute change in kinetic energy due to advection of momentum.

$$UENG_{i,k} = GM_{i,k} * ((-FUW_{i+1,k} * (U_{i+1,k} + U_{i,k}) + FUW_{i,k} * (U_{i,k} + U_{i-1,k}) * DXU2R_i - FVN_{i,k} * (UP_{i,k} + U_{i,k}) + FVSU_{i,k} * (U_{i,k} + UM_{i,k}))$$

$$VENG_{i,k} = GM_{i,k} * ((-FUW_{i+1,k} * (V_{i+1,k} + V_{i,k}) + FUW_{i,k} * (V_{i,k} + V_{i-1,k}) * DXU2R_i - FVN_{i,k} * (VP_{i,k} + V_{i,k}) + FVSU_{i,k} * (V_{i,k} + VM_{i,k}))$$

$ENGINT_2 = ENGINT_2 + (USAV_{i,k} * UENG_{i,k} + VSAV_{i,k} * VENG_{i,k}) * FX * DXU_i * DZ_k$

$ZUNENG_{i,2} = ZUNENG_{i,2} + UENG_{i,k} * DZ_k * HR_{i,j}$

$ZVNENG_{i,2} = ZVNENG_{i,2} + VENG_{i,k} * DZ_k * HR_{i,j}$.

Second Loop:

$UENG_{i,k} = GM_{i,k} * (-(W_{i,k} * (U_{i,k-1} + U_{i,k}) - W_{i,k+1} * (U_{i,k} + U_{i,k+1})) * DZ2RQ_{i,k}$

$VENG_{i,k} = GM_{i,k} * (-(W_{i,k} * (V_{i,k-1} + V_{i,k}) - W_{i,k+1} * (V_{i,k} + V_{i,k+1})) * DZ2RQ_{i,k}$

$ENGINT_3 = ENGINT_3 + (USAV_{i,k} * UENG_{i,k} + VSAV_{i,k} * VENG_{i,k}) * FX * DXU_i * DZ_k$

$ZUNENG_{i,3} = ZUNENG_{i,3} + UENG_{i,k} * DZ_k * HR_{i,j}$

$ZVNENG_{i,3} = ZVNENG_{i,3} + VENG_{i,k} * DZ_k * HR_{i,j}$.

c. Compute change in kinetic energy due to horizontal diffusion of momentum:

$$UENG_{i,k} = GM_{i,k} * (BBUJ * DXU2R_i * (DXT4R_{i+1} * (UB_{i+1,k} - UB_{i,k}) + DXT4R_i * (UB_{i-1,k} - UB_{i,k})) + CCUJ * (UBP_{i,k} - UB_{i,k}) + DDUJ * (UBM_{i,k} - UB_{i,k}) + GGUJ * UB_{i,k} - HHUJ * DXU2R_i * (VB_{i+1,k} - VB_{i-1,k}))$$

$$VENG_{i,k} = GM_{i,k} * (BBUJ * DXU2R_i * (DXT4R_{i+1} * (VB_{i-1,k} - VB_{i,k}) + DXT4R_i * (VB_{i-1,k} - VB_{i,k})) + CCUJ * (VBP_{i,k} - VB_{i,k}) + DDUJ * (VBM_{i,k} - VB_{i,k}) + GGUJ * VB_{i,k} + HHUJ * DXU2R_i * (UB_{i+1,k} - UB_{i-1,k}))$$

$ENGINT_4 = ENGINT_4 + (USAV_{i,k} * UENG_{i,k} + VSAV_{i,k} * VENG_{i,k}) * FX * DXU_i * DZ_k$

$ZUNENG_{i,4} = ZUNENG_{i,4} + UENG_{i,k} * DZ_k * HR_{i,j}$

$ZVNENG_{i,4} = ZVNENG_{i,4} + VENG_{i,k} * DZ_k * HR_{i,j}$.

d. Compute change in kinetic energy due to wind stress:

$UENG_{i,1} = GM_{i,1} * EEM_1 * (UOVER_i - UDIF_{i,1})$

$VENG_{i,1} = GM_{i,1} * EEM_1 * (VOVER_i - VDIF_{i,1})$

$$ENGINT_7 = ENGIT_7 + (USAV_{i,1} * UENG_{i,1} + VSAV_{i,1} * VENG_{i,1}) * FX * DXU_i * DZ_1$$

$$ZUNENG_{i,7} = ZUNENG_{i,7} + UENG_{i,1} * DZ_1 * HR_{i,j}$$

$$ZVNENG_{i,7} = ZVNENG_{i,7} + VENG_{i,1} * DZ_1 * HR_{i,j}.$$

e. Compute change in kinetic energy due to bottom drag:

$$UENG_{i,kz} = GM_{i,kz} * FFM_{kz} * (UDIF_{i,kz+1} - UDIF_{i,kz})$$

$$VENG_{i,kz} = GM_{i,kz} * FFM_{kz} * (VDIF_{i,kz+1} - VDIF_{i,kz})$$

$$ENGINT_8 = ENGINT_8 + (USAV_{i,kz} * UENG_{i,kz} + VSAV_{i,kz} * VENG_{i,kz}) * FX * DXU_i * DZ_{kz}$$

$$ZUNENG_{i,8} = ZUNENG_{i,8} + UENG_{i,kz} * DZ_{kz} * HR_{i,j}$$

$$ZVNENG_{i,8} = ZVNENG_{i,8} + VENG_{i,kz} * DZ_{kz} * HR_{i,j},$$

where KZ is equal to the index of the bottom level.

f. Compute change in kinetic energy due to vertical diffusion of momentum:

$$UENG_{i,k} = GM_{i,k} * (FXA * EEM_k * (UDIF_{i,k-1} - UDIF_{i,k}) - FXB * FFM_k * (UDIF_{i,k} - UDIF_{i,k+1}))$$

$$VENG_{i,k} = GM_{i,k} * (FXA * EEM_k * (VDIF_{i,k-1} - VDIF_{i,k}) - FXB * FFM_k * (VDIF_{i,k} - VDIF_{i,k+1}))$$

$$ENGINT_5 = ENGINT_5 + (USAV_{i,k} * UENG_{i,k} + VSAV_{i,k} * VENG_{i,k}) * FX * DXU_i * DZ_k$$

$$ZUNENG_{i,5} = ZUNENG_{i,5} + UENG_{i,k} * DZ_k * HR_{i,j}$$

$$ZVNENG_{i,5} = ZVNENG_{i,5} + VENG_{i,k} * DZ_k * HR_{i,j}$$

8. Compute new velocities (with incorrect vertical means). Add in remainder of Coriolis term i treated implicitly.

For explicit treatment:

$$UA_{i,k} = UB_{i,k} + C2DTUV * UA_{i,k}$$

$$VA_{i,k} = VB_{i,k} + C2DTUV * VA_{i,k}$$

For implicit treatment:

$$UDIF_{i,k} = (UA_{i,k} + FX * VA_{i,k}) * DETMR$$

$$VDIF_{i,k} = (VA_{i,k} - FX * UA_{i,k}) * DETMR$$

where $FX = C2DTUV * ACOR * 2.0 * OMEGA * SINEA_{i,j}$ and

$DETMR = 1.0/(1.0 + FX^2)$.

$$UA_{i,k} = UB_{i,k} + C2DTUV * UDIF_{i,k}$$

$$VA_{i,k} = VB_{i,k} + C2DTUV * VDIF_{i,k}$$

9. Determine the incorrect vertical means of the new velocities.

First, sum the new velocities multiplied by the vertical grid size of each cell in centimeter

$$SFU_i = SFU_i + UA_{i,k} * DZ_k$$

$$SFV_i = SFV_i + VA_{i,k} * DZ_k.$$

Find the means by dividing each column sum by depth (multiplying by the reciprocal):

$SFU_i = SFU_i * HR_{i,j}$

$SFV_i = SFV_i * HR_{i,j}$.

10. Subtract incorrect vertical means from newly computed velocities to get correct internal mode velocity.

11. If this is an energy timestep, compute total change of kinetic energy of internal mode:

$$ENGINT_1 = ENGINT_1 + (USAV_{i,k} * (UA_{i,k} - UB_{i,k}) + VSAV_{i,k} * (VA_{i,k} - VB_{i,k})) * FX * DXU_i,$$

where $FX = CS_j * DYU_j * DZ_k/C2DTUV$.

Begin computation of vorticity for input to CSU CRELAX:

1. Form curl of time change in vertically averaged equations:

$$ZTD_{i,j} = ((ZUN_i * DXU_i + ZUN_{i-1} * DXU_{i-1}) * CS_j - (ZUS_i * DXU_i + ZUS_{i-1} * DXU_{i-1}) * CS_{j-1})$$

$$ZTD_{i,j} = (((ZVN_i - ZVN_{i-1}) * DYU_j + (ZVS_i - ZVS_{i-1}) * DYU_{j-1} - ZTD_{i,j})* DXT2R_i * DYTR_j) * CSTR_j$$

2. If on an energy timestep, do analysis of external mode forcing:

$$ENGEXT_{ll} = ENGEXT_{ll} - P_{i,j} * (((ZVNENG_{i,ll} - ZVNENG_{i-1,ll}) * DYU_j + (ZVSENG_{i,ll} - ZVSENG_{i-1,ll}) * DYU_{j-1}) * DXT2R_i * DYTR_j - ((ZUNENG_{i,ll} * DXU_i + ZUNENG_{i-1,ll} * DXU_{i-1}) * CS_j - (ZUSENG_{i,ll} * DXU_i + ZUSENG_{i-1,ll} * DXU_{i-1}) * CS_{j-1}) * DYT2R_j * DXTR_i) * DXT_i * DYT_j$$

Transfer computed quantities to the north of the present row to be defined to the south in the computation for the next row.

3.4.4 CSC Compute Tracers

CSC Compute Tracers must compute the tracer elements of temperature and salinity and be readily adaptable to computing additional tracers.

3.4.4.1 CSU TRACER

CSU TRACER meets the requirements of CSC Compute Tracers by computing temperature and salinity and is designed to be adaptable to computing additional tracer elements.

3.4.4.1.1 CSU TRACER Design Specification/Constraints – There are no known constraints.

3.4.4.1.2 CSU TRACER Design

Input Data Elements:

/COX2/

FW1	heat above the freezing temperature
GICE	ice growth rate computed by the ice model

SHICE	growth rate of ice thickness

/FULLWD/

BUOY	energy transfer through buoyancy effects
DTABS	volume average of absolute change of temperature
ITT	timestep counter; total number of timesteps completed
KMT	number of vertical levels of the ocean at T points
MXP	if = 1, indicates second pass of a Euler backward timestep
NERGY	if = 1, indicates energy printout timestep
NTSI	number of timesteps between prints of a single line of data
TTDTOT	array of integrals on tracers
TVAR	change of variance of tracers

/ONEDIM/

CS	cosine of U,V point latitudes
CSTR	reciprocal of cosine of T point latitudes
DXT	zonal grid spacing across T boxes (between U,V points)
DXT4R	reciprocal of DXT × 4.0
DXU2R	reciprocal of DXU × 2.0
DYT	meridional grid spacing across T boxes (between U,V points)
DYTR	reciprocal of meridional grid spacing across T boxes
DYU	meridional grid spacing across U,V boxes (between T points)
DYUR	reciprocal of meridional grid spacing across U,V boxes
DZ	vertical layer thickness in centimeters
DZZ	vertical grid spacing between U,V,T points
DZZ2R	1.0/(DZZ * 2.0)

/RSTRT/

AREA1	fraction of the grid cell covered with ice
HEFF	mean ice thickness per grid cell

/RVR/

IRIVER	x component of river discharge
JRIVER)	y component of river discharge
KRIVER	z component of river discharge
RIVER	river discharge temperature

/SCALAR/

AH	coefficient of horizontal mixing of T
C2DTTS	two times the length of the timestep on T
DTTS	length of the timestep on T
GRAV	acceleration due to gravity; = 980.6 cm/s^2

/TSTOP/

CHENG	

/WORKSP1/

C2DZQ	vertical layer thickness in centimeters × 2; for future vectorization
DXT4RQ	1.0/(DXT * 4.0); for future vectorization
DXTQ	zonal grid spacing across T boxes; for future vectorization
DXU2RQ	1.0/(DXU * 2.0); for future vectorization

DXUQ	zonal grid spacing across U,V boxes; for future vectorization
DZ2RQ	1.0/(DZ * 2.0); for future vectorization
EEHQ	upper vertical mixing coefficient of T; for future vectorization
FFHQ	lower vertical mixing coefficient of T; for future vectorization
FM	masking array for T points; 0 indicates land points, 1 indicates ocean points
FMM	FM at row $J-1$
FMP	FM at row $J+1$
FVST	advective coefficient for south face of T box
RHON	density with a space and time invariant subtracted for the row to the north of the present row
RHOS	density with a space and time invariant subtracted for the row to the south of the present row
T	tracer variables for current timestep, row J
TB	tracer for timestep before present
TBM	similar to TB but in row $J-1$
TBP	similar to TB but in row $J+1$
TM	tracer in row $J-1$ for present timestep
TP	tracer in row $J+1$ for present timestep
UM	zonal component of velocity in row $J-1$
V	meridional component of velocity

Output Data Elements:

/COX2/

FW1	fraction of the grid cell covered by ice
GICE	ice growth rates computed by the ice portion of the PIPS2.0 coupled model

/FULLWD/

BUOY	energy transfer through buoyancy effects
DTABS	volume average of absolute change of temperature
TTDTOT	array of integrals on tracers
TVAR	change of variance of tracers

/WORKSP1/

FVST	advective coefficient for south face of T box
RHOS	density with a space and time invariant subtracted for the row to the south of the present row
T	tracer variables (temperature and salinity) for current timestep, row J
TA	tracer variables for timestep after present
TDIF	diffusion computation array
TEMPA	temporary storage
TEMPB	temporary storage
UA	zonal component of velocity for timestep after present
VA	meridional component of velocity for timestep after present

Parameters:

stored in file ocean.par

The ocean.par parameters are described in Sec. 3.4.1.1.2.

Logic Flow:

Prepare arrays for the computation of the tracers:.

Several arrays must be calculated in order to prepare for the computation of the tracers:

- the advective coefficients for the west (*FUW*) and north (*FVN*) faces of the T box,
- vertical velocity in T columns, and
- boundary conditions for the vertical diffusion of tracers.

1. Find advective coefficients for west and north faces of T box:

   $FUW_{i,k} = (U_{i-1,k} * DYU_j + UM_{i-1,k} * DYU_{j-1}) * FXA$

   $FVN_{i,k} = (V_{i,k} * DXUQ_{i,k} + V_{i-1,k} * DXUQ_{i-1,k}) * FXB * DXT4RQ_{i,k}.$

2. Compute vertical velocity in T columns:

   a. Set vertical velocity at the surface to 0 (rigid–lid assumption).

   b. Compute change of vertical velocity between levels:

      $W_{i,k+1} = C2DZQ_{i,k} * ((FUW_{i+1,k} - FUW_{i,k}) * DXT4RQ_{i,k} + FVN_{i,k} - FVST_{i,k}).$

   c. Integrate downward from the surface by adding successive vertical velocities.

3. Set boundary conditions for vertical diffusion of tracers:

   a. Transfer interior points (*TB*) into diffusion computation array (*TDIF*).

   b. Set top point of the column to reflect surface flux, bottom point of the column to refle insulation:

      $TDIF_{i,1,m} = TB_{i,1,m}$

      $TDIF_{i,kz+2,m} = TB_{i,kz,m}.$

Compute the tracers:

Implement the equations given in Sec. 2.2.4.3 for computing tracer elements. Two passes a required: in the first pass, temperature is computed; in the second pass, salinity is computed.

1. Compute total advection of tracers by summing the flux through the west face of the T bo the zonal flux divergence, the meridional flux divergence, the flux through the top of t T box, and the vertical flux divergence.

   a. Compute flux through west face of T box:

      $TEMPA_{i,k} = FUW_{i,k} * (T_{i,k,m} + T_{i-1,k,m})$

      $TA_{i,1,1} = MAX(TA_{i,1,1} - 54.4 * (TA_{i,1,2} + 0.035))$

      $TMPCHG_{i,k} = FUW_{i,k} * T_{i,k,m} + T_{i-1,k,m} + 2.0 * TFRG.$

   b. Compute zonal flux divergence for all grid cells in slab *j*:

      $TA_{i,k,m} = (TEMPA_{i,k} - TEMPA_{i+1,k}) * DXT4RQ_{i,k}.$

      For surface temperature,

      $TCHENG_{i,j} = (TMPCHG_{i,k} - TMPCHG_{i+1,k}) * DXT4RQ_{i,k}.$

c. Add in meridional flux divergence:

$TA_{i,k,m} = TA_{i,k,m} - FVN_{i,k} * (TP_{i,k,m} + T_{i,k,m}) + FVST_{i,k} * (T_{i,k,m} + TM_{i,k,m})$

For surface temperature,

$TCHENG_{i,j} = TCHENG_{i,j} - FVN_{i,k} * (TP_{i,k,m} + T_{i,k,m}) + FVST_{i,k} * (T_{i,k,m} + TM_{i,k,m})$.

d. Compute flux through top of T box:

$TEMPB_{i,k} = W_{i,k} * (T_{i,k-1,m} + T_{i,k,m})$.

For surface temperature,

$TMPCHG_{i,k} = W_{i,k} * (T_{i,k-1,m} + T_{i,k,m} + 2.0 * TFRG)$.

e. Add in vertical flux divergence:

$TA_{i,k,m} = TA_{i,k,m} + (TEMPB_{i,k+1} - TEMPB_{i,k}) * DZ2RQ_{i,k}$

For surface temperature,

$TCHENG_{i,j} = TCHENG_{i,j} + (TMPCHG_{i,k+1} - TMPCHG_{i,k}) * DZ2RQ_{i,k}$.

2. Add horizontal diffusion of tracers (evaluated at tau – 1 timestep) to the total advection of the tracers.

a. Compute coefficients dependent only on latitude:

$BBTJ = 8.0 * AH * CSTR_j^2$

$CCTJ = AH * CS_j * DYUR_j * DYTR_j * CSTR_j$

$DDTJ = AH * CS_{j-1} * DYUR_{j-1} * DYTR_j * CSTR_j$

b. Compute gradients at west face of T box:

$TEMPA_{i,k} = DXU2RQ_{i-1,k} * (TB_{i,k,m} - TB_{i-1,k,m})$.

c. Add in final contribution from horizontal diffusion of tracers. To provide for insulated walls, each gradient is multiplied by the mask of the point in its respective direction causing it to be 0 if it is taken across a wall:

$$TA_{i,k,m} = TA_{i,k,m} + BBTJ * DXT4RQ_{i,k} * (FM_{i+1,k} * TEMPA_{i+1,k} - FM_{i-1,k} * TEMPA_{i,k}) + CCTJ * FMP_{i,k} * (TBP_{i,k,m} - TB_{i,k,m}) + DDTJ * FMM_{i,k} * (TBM_{i,k,m} - TB_{i,k,m}).$$

3. Add vertical diffusion of tracers to the sum of the total advection of the tracers plus the horizontal diffusion of the tracers.

a. Compute gradients at top of T box:

$TEMPB_{i,k} = TDIF_{i,k,m} - TDIF_{i,k+1,m}$.

b. Add in final contribution from vertical diffusion of tracers:

$TA_{i,k,m} = TA_{i,k,m} + EEHQ_{i,k} * TEMPB_{i,k} - FFHQ_{i,k} * TEMPB_{i,k+1}$.

4. Set Newtonian boundary condition of temperature and salinity at the ocean surface.

   For surface grid cells:

   $TA_{i,1,m} = TA_{i,1,m} + 4.63\text{E}{-8} * (CHENG_{i,j,m,1} - TB_{i,1,m})$.

   For surface temperature (m=1):

   $TA_{i,1,1} = TA_{i,1,1} - GICE_{i,j} * RICE0/30.00$,

   where $RICE0 = 302.0\text{E}6/4.19\text{E}6$.

   For surface salinity, in open water, the atmospheric heating should not increase or decrease the salinity because no ice melting is involved. Open-water grid cells are indicated by $AICE(i,j,2) \leq 0.15$, $HICE(i,j,2) \leq 0.1$, and $SHICE(i,j) \leq 0$. For open-water grid cells (surface level only) reset ASH to 0.

   For surface salinity:

   $TA_{i,1,2} = TA_{i,1,2} + SHICE * 0.035 * ASH/30.00$.

   For all other levels ($k \neq 1$):

   $TA_{i,k,m} = TA_{i,k,m} + 4.63\text{E}{-8} * (CHENG_{i,j,m,k} - TB_{i,k,m})$.

5. Compute new tracers, resetting land points to 0 by multiplying by the land mask array, *FM*. New tracers are computed by multiplying the time rate of change of tracer values (computed in steps 1–4 above and represented by *TA*) by twice the timestep on T and adding this value to the tracer quantities computed for the timestep before present. Factor the *FM* array into the equation to force 0 values for array elements that represent land points.

6. Set salinity to 45 ppt over land to stop convection there. $TA_{i,k,2} = 0.01$ is equal to 45 ppt, since model units are (ppt – 35.0)/1000.0.

7. On an energy timestep, perform analysis of tracer forcing.

   a. Compute change of tracer due to advection:

   $$TTDTOT_{2,m} = TTDTOT_{2,m} + BOXVOL * ((-FUW_{i+1,k} * (T_{i+1,k,m} + T_{i,k,m}) + FUW_{i,k} * (T_{i,k,m} + T_{i-1,k,m})) * DXT4R_i - FVN_{i,k} * (TP_{i,k,m} + T_{i,k,m}) + FVST_{i,k} * (T_{i,k,m} + TM_{i,k,m}))$$

   b. Compute change of tracer due to horizontal diffusion:

   $$TTDTOT_{4,m} = TTDTOT_{4,m} + BOXVOL * (BBTJ * DXU2R_i * DXT4R_i * FM_{i+1,k} * (TB_{i+1,k,m} - TB_{i,k,m}) + BBTJ * DXU2R_{i-1} * DXT4R_i * FM_{i-1,k} * (TB_{i-1,k,m} - TB_{i,k,m}) + CCTJ * FMP_{i,k} * (TBP_{i,k,m} - TB_{i,k,m}) + DDTJ * FMM_{i,k} * (TBM_{i,k,m} - TB_{i,k,m}))$$

   c. Compute change of tracer due to vertical diffusion:

   $$TTDTOT_{5,m} = TTDTOT_{5,m} + BOXVOL * (EEH_k * (TDIF_{i,k,m} - TDIF_{i,k+1,m}) - FFH_k * (TDIF_{i,k+1,m} - TDIF_{i,k+2,m}))$$

   d. Add in contribution from Newtonian boundary condition of temperature and salinity at the ocean surface.

For surface temperature:

$$TTDTOT_{5,1} = TTDTOT_{5,1} + 4.63E{-}8 * (CHENG_{i,j,m,k} - TB_{i,k,m}) * DZ_k \\ GICE_{i,j} * RICE0 * DZ_k/30.00.$$

For surface salinity:

$$TTDTOT_{5,2} = TTDTOT_{5,2} + 4.63E{-}8 * (CHENG_{i,j,m,k} - TB_{i,k,m}) * DZ_k \\ + SHICE_{i,j} * 0.035 * ASH * DZ_k/30.00,$$

for other than surface grid cells ($k \neq 1$).

For temperature:

$$TTDTOT_{5,1} = TTDTOT_{5,1} + 4.63E{-}8 * (CHENG_{i,j,m,k} - TB_{i,k,m}) * DZ_k.$$

For salinity:

$$TTDTOT_{5,2} = TTDTOT_{5,2} + 4.63E{-}8 * (CHENG_{i,j,m,k} - TB_{i,k,m}) * DZ_k.$$

e. Compute total energy exchange between potential and kinetic:

$$BUOY = BUOY - FX * DZZ_k * W_{i,k} * (RHOS_{i,k-1} + RHOS_{i,k})$$

8. Convectively adjust water column if gravitationally unstable. Allow for variable number of passes through convection loop using an outer loop with limits from 1 to *NCON* where *NCON* is presently hardwired to 1.

   a. Call CSU STATEC to compute density for entire slab to determine stability.

   b. For each tracer, mix adjoining levels if unstable. Tracers are unstable if density of a grid cell is greater than the density of the grid cell below it. To mix unstable grid cells:

   $$TA_{i,k,m} = (DZ_k * TA_{i,k,m} + DZ_{k+1} * TA_{i,k+1,m}) * DZZ2R_{k+1}.$$

Integrate changes and prepare for next timestep:

1. Integrate total changes in T, S and squared T, S on energy timestep.

2. Accumulate integrated absolute changes in T every *NTSI* timesteps.

3. Transfer quantities computed to the north of the present row to be defined to the south in the computation of the next row.

4. Set new velocities at northern wall to 0 since no pass through CSU CLINIC is made for this row.

3.4.5 CSC Compute External Mode

The external mode of velocity must be computed in terms of a mass transport stream function.

3.4.5.1 CSU CRELAX

CSU CRELAX is called once at the end of each timestep by CSU OSTEP. It takes the vorticity function computed in CSU CLINIC and, using sequential overrelaxation, solves the LaPlacian equation for the external mode of velocity in terms of a mass transport stream function.

3.4.5.1.1 CSU CRELAX Design Specification/Constraints – There are no known constraints.

4.4.5.1.2 CSU CRELAX Design

Input Data Elements:

/FIELDS/

HR	reciprocal of total depth at U,V point
P	mass transport stream function
PB	mass transport stream function for previous timestep
ZTD	change of vorticity across one timestep

/FULLWD/

EB	if true: the current step is a Euler backward timestep
IEIS	array of *I* ending indices for island boxes
IEZ	array of ending indices for vorticity
ISIS	array of *I* starting indices for island boxes
ISZ	array of starting indices for vorticity
ITT	timestep counter
JEIS	array of *J* ending indices for island boxes
JSIS	array of *J* starting indices for island boxes
KFLDS	disk unit number for two-dimensional horizontal fields and start and en indices; = 12
MIX	mixing timestep indicator; set equal to 1 on a mixing timestep
MXP	if = 1, second pass of a Euler backward timestep
MXSCAN	maximum number of scans allowed for convergence

/ONEDIM/

CS	cosine of U,V point latitudes
CST	cosine of T point latitudes
CSTR	reciprocal of cosine of T point latitudes
DXT	zonal grid spacing across T boxes
DXTR	1.0/DXT
DXUR	1.0/DXU
DYT	meridional grid spacing across T boxes
DYTR	reciprocal of meridional grid spacing across T boxes (between U,V points
DYUR	reciprocal of meridional grid spacing across U,V boxes (between T points
SINEA	sine of U,V point latitude in the Earth-oriented coordinates; used to comput Coriolis force

/SCALAR/

ACOR	if > 0, treat the Coriolis term implicitly with forward component weighte by ACOR, past component by 1 – ACOR
C2DTSF	DTSF × 2
CRIT	criterion for convergence of relaxation
OMEGA	rate of rotation of the coordinate system
SOR	coefficient of overrelaxation

Output Data Elements:

/FIELDS/

P	mass transport stream function
PB	mass transport stream function for previous timestep

/FULLWD/

MSCAN	scan counter

/WORKSP2/

CFE	coefficient of eastern point in LaPlacian star
CFN	coefficient of northern point in LaPlacian star
CFS	coefficient of southern point in LaPlacian star
CFW	coefficient of western point in LaPlacian star
COF	normalization array in computation of island flow
COFIS	integral of COF
CPF	normalization factor used in constructing LaPlacian star
ISMASK	gridpoint type indicator: = 0 over interior points; = 1 over perimeter points; = 2 over land points
PTD	change of stream function across a timestep
PTDB	change of stream function across previous timestep
RES	residual of relaxation

Parameters:
stored in file ocean.par

The ocean.par parameters are described in Sec. 3.4.1.1.2.

Logic Flow:
Prepare for the relaxation:

Begin introductory section to prepare for the relaxation:

1. Initialize working arrays.

2. Read in the relaxation solution from the timestep before present and the relaxation solution of the present timestep.

3. Form island mask by distinguishing interior ocean points, perimeter ocean points, and land points.

4. Calculate the depth field from the depth field reciprocal array.

5. Ensure that all points over land are exactly 0.

6. Generate arrays of coefficients for relaxation.

   a. Compute coefficients of the LaPlacian star, augment coefficients for implicit treatment of Coriolis term, and normalize.

   Northern point correction coefficient:

   $CFN_{i,j} = 2.0 * CS_j * CSTR_j * DYTR_j * DYUR_j/(H_{i-1,j} + H_{i,j})$

   $CFN_{i,j} = CFN_{i,j} + (HR_{i,j} - HR_{i-1,j}) * SINEA_{i,j} * FX * DXTR_i$

   $CFN_{i,j} = CFN_{i,j} * CPF_i.$

   Southern point correction coefficient:

   $CFS_{i,j} = 2.0 * CS_{j-1} * CSTR_j * DYTR_j * DYUR_{j-1}/(H_{i-1,j-1} + H_{i,j-1})$

$$CFS_{i,j} = CFS_{i,j} - (HR_{i,j-1} - HR_{i-1,j-1}) * SINEA_{i,j-1} * FX * DXTR_i$$

$$CFS_{i,j} = CFS_{i,j} * CPF_i.$$

Eastern point correction coefficient:

$$CFE_{i,j} = 2.0 * CSTR_j^2 * DXUR_i * DXTR_i/(H_{i,j} + H_{i,j-1})$$

$$CFE_{i,j} = CFE_{i,j} - (HR_{i,j} * SINEA_{i,j} - HR_{i,j-1} * SINEA_{i,j-1}) * FX * DXTR_i$$

$$CFE_{i,j} = CFE_{i,j} * CPF_i.$$

Western point correction coefficient:

$$CFW_{i,j} = 2.0 * CSTR_j^2 * DXUR_{i-1} * DXTR_i/(H_{i-1,j} + H_{i-1,j-1})$$

$$CFW_{i,j} = CFW_{i,j} + (HR_{i-1,j} * SINEA_{i,j} - HR_{i-1,j-1} * SINEA_{i,j-1}) * FX * DXTR_i$$

$$CFW_{i,j} = CFW_{i,j} * CPF_i,$$

where $FX = -C2DTSF * ACOR * CSTR_j * DYTR_j * OMEGA$.

b. Compute coefficients on island perimeter points. Identify islands according to start and sto island indices stored in ISIS, IEIS, JSIS, and JEIS for ISMASK array. A value of 1 is use to indicate island perimeter points in the ISMASK array.

Island coefficients are computed based on which side of the grid cell is adjacent to lan (identified by $HR = 0$).

Assign northern coefficient if $HR_{i-1,j}$ is not 0 or $HR_{i,j}$ is not 0:

$$CFN_{i,j} = (2.0 * CS_j * DYUR_j * DYTR_j * CSTR_j)/(PTD_{i-1,j} + PTD_{i,j}) + FX * DXTR_i * (HR_{i,j} - HR_{i-1,j}) * SINEA_{i,j}.$$

Assign southern coefficient if $HR_{i-1,j-1}$ is not 0 or $HR_{i,j-1}$ is not 0:

$$CFS_{i,j} = FX/(PTD_{i-1,j-1} + PTD_{i,j-1}) - FXD * DXTR_i * (HR_{i,j-1} - HR_{i-1,j-1}) * SINEA_{i,j-1}.$$

Assign eastern coefficient if $HR_{i,j}$ is not 0 or $HR_{i,j-1}$ is not 0:

$$CFE_{i,j} = FXA * DXTR_i * DXUR_i/(PTD_{i,j} + PTD_{i,j-1}) - FXD * DXTR_i * (HR_{i,j} * SINEA_{i,j} - HR_{i,j} * SINEA_{i,j-1}).$$

Assign western coefficient if $HR_{i-1,j}$ is not 0 or $HR_{i-1,j-1}$ is not 0:

$$CFW_{i,j} = FXA * DXTR_i * DXUR_{i-1}/(PTD_{i-1,j} + PTD_{i-1,j-1}) + FXD * DXTR_i * (HR_{i-1,j} * SINEA_{i,j} - HR_{i-1,j-1} * SINEA_{i,j-1}).$$

Multiply all island perimeter coefficients *CFN*, *CFS*, *CFE*, and *CFW* by *COF* normalizati array.

Sum *COFIS* over all island grid boxes.

c. Multiply all of the *CFN*, *CFS*, *CFE*, and *CFW* coefficients by the overrelaxation factor SC read in from NAMELIST PARMS. Also set *COFIS* for the island equal to its reciproca

7. Compute a first guess for the relaxation by extrapolating the two previous solutions forward in time:

   a. Finish reading in the relaxation solution of the previous timestep, array PTD.

   b. Perform time extrapolation, accounting for the mixing timestep.

   $PTD_{i,j} = FXA * (2.0 * PTD_{i,j} - PTDB_{i,j})$,

   where $FXA = 0.5$ on a mixing timestep and 1.0 otherwise.

8. Compute criterion for the convergence of relaxation: $CRTP = CRIT * FXA * SOR$.

9. Initialize residuals array *RES* to 0.

Loop to compute the relaxation:

This algorithm uses the method of successive overrelaxation where a guess is made for all grid cells variant in *P* (the stream function), a residual is computed based on the stream function and the boundary conditions, and a new guess is established from the residual. This process is iterated until the change in *P* between guesses is less than the specified criterion.

1. Compute entire field of residuals as in simultaneous relaxation:

   $$RES_{i,j} = CFN_{i,j} * PTD_{i,j+1} + CFS_{i,j} * PTD_{i,j-1} + CFE_{i,j} * PTD_{i+1,j} + CFW_{i,j} * PTD_{i-1,j} - SOR * (PTD_{i,j} + ZTD_{i,j}).$$

2. Reset residuals over land to 0 based on start and end indices in arrays ISZ and IEZ.

3. Perform correction on southern point to yield sequential relaxation using correction factor computed in the initialization procedure:

   $RES_{i,j} = RES_{i,j} + CFS_{i,j} * RES_{i,j-1}$.

4. Perform correction on western point to yield sequential relaxation using correction factor computed in the initialization procedure:

   $RES_{i,j} = RES_{i,j} + CFW_{i,j} * RES_{i,j-1}$.

5. Correct change of stream function based on residuals by adding $RES_{i,j}$ to $PTD_{i,j}$.

6. Find the maximum absolute residual to determine convergence.

7. Do hole relaxation for each island and loop for the number of islands in the model basin.

   a. Compute island residuals:

   Initialize island residual (*RESIS*) to 0.

   Loop for the meridional start to stop island grid boxes (index *J*).

   Loop for the zonal start to stop island grid boxes (index *I*).

   If $ISMASK_{i,j}$ equals 1, increment RESIS for this grid box:

   $$RESIS = RESIS + (CFN_{i,j} * PTD_{i,j+1} + CFS_{i,j} * PTD_{i,j-1} + CFE_{i,j} * PTD_{i+1,j} + CFW_{i,j} * PTD_{i-1,j} - SOR * (PTD_{i,j} + ZTD_{i,j})) * COF_{i,j}.$$

b. Normalize the island residual by its coefficient COFIS and update the maximum absolute residual of the relaxation if necessary.

c. Correct change of stream function over the island and its perimeter points by adding the island residual to it.

8. Test the maximum residual for convergence of the relaxation. If not converged, proceed with another scan by returning to step 1. If the maximum number of scans has been reached, accept the solution and proceed to update the stream function.

Update the stream function based on the relaxation solution:

1. Update the stream function based on the relaxation solution. On the second pass of the Euler backward timestep, update only the stream function for the present timestep *P*, since the stream function for the timestep before present *PB* was updated on the first pass. If the current timestep is not the second pass of a Euler backward timestep, update the stream functions for the present timestep and the timestep before present.

2. Save change of stream function to compute first guess for relaxation next timestep. Bypass this section on the first pass of a Euler backward timestep, since it will be done on the second pass. Multiply the change of stream function by 2 if on a mixing timestep or if on the second pass of a Euler backward timestep.

3.4.6 CSC Direct Access Manager

The ocean model requires a set of routines that interface it with the local system I/O facility.

3.4.6.1 CSU ODAM

CSU ODAM consists of several entry points that collectively fulfill the requirements of CSC Direct Access Manager.

3.4.6.1.1 CSU ODAM Design Specification/Constraints – CSU ODAM will consist of several entry points instead of subroutines because of the close relationships between the I/O routines. For the core-contained mode, the routines contained in CSU ODAM must manage the transfer of data as if they resided in different files. In actuality, data for five logical units will all be stored in the same array (virtual disk). Logical units LABS(1), LABS(2), and LABS(3) (13,14, and 15, respectively) will be used for primary slab data. Logical unit KFLDS (set to 12) will be used for two-dimensional horizontal fields and logical unit KONTRL (set to 11) will be used for the timestep counter, total elapsed time, and the area and volume of the basin. Data stored in each of these units will be stored in an array in contiguous slots and in ascending order on logical unit number (e.g., data for unit 11 will be located in the first slot, data for unit 12 in the second, and so forth).

3.4.6.1.2 CSU ODAM Design

Input Data Elements:

A	origination/destination array in memory
LO	I/O unit number
LU	I/O unit number
NBLK	length of each block on the unit
NBUF	number of buffers supplied to the unit
NFRST	unit address of the first word to be transferred
NTOT	length of unit in words

NWRS	number of words to transfer
BIG	read from tape to initialize virtual disk and later stored to tape to save contents of virtual disk

Output Data Elements:

A	origination/destination array in memory
BIG	read from tape to initialize virtual disk and later stored to tape to save contents of virtual disk

Contents of array BIG and more detailed ODAM I/O virtual disk access information is contained in Sec. 3.4.6.

Local Data Elements:

N11	20
N12	907200
NE	address of the last word to retrieve from array C
NS	address of the first word to retrieve from array C
NSKP	number of words to skip on retrieval from array C
NSL	8035200
NTB	16977620

Parameters:

stored in file ocean.par

The ocean.par parameters are described in Sec. 3.4.1.1.2.

stored in file odam.par

N11	20
N12	907200
NSL	8035200
NTB	16977620

Logic Flow:

Define several entry points to simulate direct access input/output:

ENTRY OGET

ENTRY OGET will retrieve a variable number of words from array *C* and store them in array *A*. For $LU > 11$, $N11$ words are skipped (represented by *NSKP*). For $LU > 12$, $N12$ additional words are skipped. For the first word retrieved (*NS*), $NS = NFRST + NSKP$ for logical units = 11 or 12. The first word retrieved for logical units ≥ 13 is computed as follows:

$$NS = NSKP + (2 * ((NFRST - 1)/NWRS) + (LU - 13)) * NWRS + 1.$$

The last word retrieved $(NE) = NS + NWRS - 1$. In a core-contained mode, data is retrieved from array *C* for elements *NS* through *NE* and assigned to array *A* to be sent back to the calling routine.

ENTRY OPUT

ENTRY OPUT transfers a variable number of words from array *A* in memory and stores them in array *C* on virtual disk. The starting location for data storage is first determined by the logical unit number *LU* and the variable *NFRST* passed in the argument list. For logical unit numbers >11,

the number of words to skip (*NSKP*) is initialized to *N*11. For logical unit numbers > 12, th number of words to skip is incremented by an additional *N*12. The start location for data storag (*NS*) for logical units < 13 = *NFRST* + *NSKP*. The start location for data storage for logical units ≥ 1 is computed as follows:

$$NS = NSKP + (2 * ((NFRST - 1)/NWRS) + (LU - 13)) * NWRS + 1.$$

The ending location for data storage (*NE*) = *NS* + *NWRS* – 1. Contiguous data elements beginnin with the first word of array *A* are stored in array *C* in locations *NS* through *NE*.

ENTRY ORD

ENTRY ORD initializes the virtual disk from tape. The array BIG (equivalenced to *C*) is rea from the tape drive logical unit number *LO*.

ENTRY OWRT

ENTRY OWRT saves the virtual disk data to tape. Array BIG (equivalenced to *C*) is stored o the tape drive logical unit number *LO*.

ENTRY OCLOSE

In core-contained mode, ENTRY OCLOSE does not perform any function but serves as placeholder for future conversion to actual disk I/O.

ENTRY OFIND

In core-contained mode, ENTRY OFIND does not perform any function but serves as a place holder for future conversion to actual disk I/O.

3.5 CSC Next Timestep

The CSC Next Timestep is used after each timestep run of CSCs Ice and Ocean to upda timestep information. Its CSU SWAP calculates a new date-time group, new currents, and ne mixed-layer temperatures for the next pass through the ice and ocean models.

3.5.1 CSU SWAP

CSU SWAP meets the requirements of CSC Next Timestep for the next timestep. It also swa positions of currents, temperatures and salinities and calculates a new mixed-layer temperature

3.5.1.1 CSU SWAP Design Specification/Constraint

There are no known constraints.

3.5.1.2 CSU SWAP Design

Input Data Elements:

/COX2/

FW1	heat above the freezing temperature
GICE	ice growth rate computed by the ice model
TM1	previous timestep mixed-layer temperature
TM2	previous two timestep mixed-layer temperature

/CURNTS/

T0	present timestep of temperature and salinity
T1	previous timestep of temperature and salinity
T2	two timesteps ago of temperature and salinity
U0	present timestep of x component of ocean current
U1	previous timestep of x component of ocean current
V0	present timestep of y component of ocean current
V1	previous timestep of y component of ocean current

/FULLWD/

NLAST	final timestep to compute on this run of model

/OCEANS/

FW	oceanic heat flux

/TSTEP/

IDTG	date-time group; read from standard input (YYMMDDHH)
MDY	day calculated from date-time group
MHR	hour calculated from date-time group
MM	month calculated from date-time group
MYR	year calculated from date-time group

Output Data Elements:

/COX2/

GICE	ice growth rate computed by the ice model
TFRZ	temperature at freezing point
TM1	previous timestep mixed-layer temperature
TM2	previous two timestep mixed-layer temperature

/CURNTS/

T1	previous timestep of temperature and salinity
T2	two timesteps ago of temperature and salinity
U1	previous timestep of x component of ocean current
V1	previous timestep of y component of ocean current

/FULLWD/

NFIRST	restart indicator; = 1 to start from scratch; = 0 to restart from data
NLAST	final timestep to compute on this run of model
NNERGY	10 timesteps past NLAST

/OCEANS/

FW	oceanic heat flux

/RFOR2/

GWATX	x component of the ocean current
GWATY	y component of the ocean current

/TSTEP/

IDTG	date-time group; read from standard input (YYMMDDHH)
MDY	day calculated from date-time group
MHR	hour calculated from date-time group
MM	month calculated from date-time group
MYR	year calculated from date-time group

Local Data Elements:

CP	constant of water heat capacity	real
DELTAT	timestep in seconds	real
EXCHNG	number of times to exchange ice-ocean data	real
I	index counter	integer
IHR	timestep in hours	integer
IMON	index counter for obtaining current month	integer
J	index counter	integer
KK	index counter	integer
RLATNT	constant of water latent heat	real
TDAY	timestep to exchange ice-ocean data	real
ZMIX	depth of mixed layer (30 m)	real

Logic Flow:

The next timesteps' date-time group, year, month, day, and hour values are calculated. The timestep counters are updated. Previous timestep temperature, salinity, and current fields are moved one timesteps backward. Present timestep fields are moved to the previous timestep fields. New seawater freezing temperatures are calculated according to salinity as –54.4 * salinity. Ice thickness growth rate of open water, the total ice thickness growth rate, and the heat above the freezing temperature from the present timestep are swapped with the previous timestep. If the growth rate is greater than that of the heat above freezing temperature, and the heat above freezing is > 0 the growth rate is set to 0 (no more cooling). A new heat flux is calculated and its value is divided by 118.

Algorithms:

The new oceanic heat flux is calculated using:

$$Q = c_p \; z_m \; (T_{m_i} - T_{m_{i-1}})/t_d/n_t + GL_w,$$

where c_p is the constant of water heat capacity,

z_m is the depth of the mixed layer,

T_{m_i} is the mixed-layer temperature from the previous timestep,

$T_{m_{i-1}}$ is the mixed-layer temperature from two timesteps ago,

t_d/n_t is the timestep on which to exchange data (t_d is the number of seconds in a day and n_t is the number of times to exchange data),

G is the ice growth rate, and

L_w is the constant of water latent heat.

3.6 CSC Output

The CSC Output is the last CSC executed in PIPS2.0. It is responsible for writing the data file containing the ice model and ocean model restart data. These restart values are used by PIPS2.0 for next model run.

3.6.1 CSU SAV_ICE

The purpose of the CSU SAV_ICE is to write the ice model restart fields to a file.

3.6.1.1 CSU SAV_ICE Design Specification/Constraint

The ice model restart data are written to a file connected to unit 33.

3.6.1.2 CSU SAV_ICE Design

Input Data Elements:

/RSTRT/

AREA1	fraction of the grid cell covered by thick ice
HEFF	mean ice thickness per grid cell
TICE	mixed-layer temperature for open water or ice temperature for ice cover
UICE	x component of the ice drift
UICEC	intermediate x component of the ice drift
VICE	y component of the ice drift
VICEC	intermediate y component of the ice drift

Output Data Elements:

UICE	x component of the ice drift
UICEC	intermediate x component of the ice drift
VICE	y component of the ice drift
VICEC	intermediate y component of the ice drift
HEFF	mean ice thickness per grid cell
AREA1	fraction of the grid cell covered by thick ice
TICE	mixed-layer temperature for open water or ice temperature for ice cover

Parameters:

stored in file ice.par

The ice.par parameters are described in Sec. 3.1.1.2.

Data element design information is provided in Sec. 4.0.

Logic Flow:

The SAV_ICE CSU is called by CSU DRIVER.

All restart fields are written to a file connected to unit 33. These restart fields include x and y components of the ice drift (UICE and VICE), intermediate x and y components of the ice drift (UICEC and VICEC), mean ice thickness per grid cell (HEFF), fraction of the grid cell covered by thick ice (AREA1), and mixed-layer temperature for open water or ice temperature for ice cover (TICE).

Local data files or database:

The restart data are written to the 92<*mmdd*>.res (unit 33) file, where <*mmdd*> is the month and day of the current day. This file is a permanent, binary file that contains the x and y components of the ice drift, the intermediate x and y components of the ice drift, the mean ice thickness per grid cell, the fraction of the cell covered by thick ice, and the mixed-layer temperature in the case of open water or the ice temperature in the case of an ice cover. The data in this file are used to restart a run.

3.6.2 CSU SAV_OCN

The purpose of the CSU SAV_OCN is to write the ocean model restart fields to a file.

3.6.2.1 CSU SAV_OCN Design Specification/Constraint

The ocean model restart data are written to a file connected to unit 30.

3.6.2.2 CSU SAV_OCN Design

Input Data Elements:

/COX2/

FW1	heat above the freezing temperature
GICE	ice growth rate computed by the ice model
SHICE	total ice thickness

/CURNTS/

T0	present timestep of temperature and salinity
T1	previous timestep of temperature and salinity
U0	present timestep of x component of ocean current
U1	previous timestep of x component of ocean current
V0	present timestep of y component of ocean current
V1	previous timestep of y component of ocean current

/FULLWD/

ITT	timestep counter

Output Data Elements:

ITT	timestep counter
U0	present timestep of x component of ocean current
V0	present timestep of y component of ocean current
T0	present timestep of temperature and salinity
U1	previous timestep of x component of ocean current
V1	previous timestep of y component of ocean current
T1	previous timestep of temperature and salinity

Parameters:

stored in file ice.par

The ice.par parameters are described in Sec. 3.1.1.2.

Local Data Elements:

I	index counter	integer
J	index counter	integer
M	index counter	integer

Logic Flow:

The CSU SAV_OCN is called by CSU DRIVER.

All restart fields are written to a file connected to unit 30. The restart fields include the i growth rate, present and previous timesteps of temperature and salinity, and the present and previo timesteps of the x and y components of ocean current.

Local data files or database:

The restart data are written to the for010_<*mmdd*>.dat (unit 30) file, where <*mmdd*> is t month and day of the current day. This file is a permanent, binary file. The data in this file is us to restart the next model run.

4.0 PIPS2.0 DATA TABLES

Data Tables are presented for the two higher level CSCs: CSC Compute Ice and CSC Compu Ocean.

4.1 Ice Data Table

This section describes the global data elements of CSC Compute Ice.

NAME	DATA TYPE	DESCRIPTION	CSU DEFINED IN	CSU USED IN	COMMON BLOCK
A22	real	minimal compactness allowed defined by data statement; (0.15)	ICEMDL, HEAT	GROWTH, HEAT, ICEMDL	local variable
AMASS	real	ice mass per grid area defined by data statement; array with dimensions (*IMT,JMT*)	FORM	FORM, ICEMDL, RELAX	local variable
AREA1	real	fraction of grid cell covered by ice; read from restart data file or initialized to constant initial condition; array with dimensions (*IMT,JMT*,3)	RST_ICE	DRIVER, FORM, GROWTH, HEAT, ICEMDL, RST_ICE, SAV_ICE, SSMI, STRESSUP, TRACER	RSTRT
CFO	real	heat above freezing in terms of ice thickness growth rate or ice thickness growth rate in open water; array with dimensions (*IMT,JMT*)	HEAT	HEAT	COX2
DELTAT	real	timestep (s)	ICEMDL	ADVECT, BUDGET, GROWTH, ICEMDL, RELAX, SWAP	STEP
DELTAX	real	x (longitude) grid spacing (deg)	ICEMDL, SWAP	ADVECT, DIFFUS, ICEMDL, PLAST, SWAP	STEP
DELTAY	real	y (latitude) grid spacing (deg)	ICEMDL, SWAP	ADVECT, DIFFUS, ICEMDL, PLAST, RELAX, SWAP	STEP
DELTT	real	timestep based on type of timestep differencing (h)	ICEMDL	ADVECT, DIFFUS, ICEMDL	local variable
DELTXA	real	x (longitude) grid spacing for thermodynamic fields (m); array with dimensions (*JMT*)	ICEMDL	ADVECT, DIFFUS, PLAST, ICEMDL	STEPSP
DELTXU	real	x (longitude) grid spacing for velocity fields (m); array with dimensions (*JMTM1*)	ICEMDL	FORM, GEOWIND, ICEMDL, RELAX	STEPSP

NAME	DATA TYPE	DESCRIPTION	CSU DEFINED IN	CSU USED IN	COMMON BLOCK
DELTYA	real	y (latitude) grid spacing for thermodynamic fields (m)	ICEMDL	ADVECT, DIFFUS, ICEMDL, PLAST	STEPSP
DELTYU	real	y (latitude) grid spacing for velocity fields (m)	ICEMDL	FORM, GEOWIND, ICEMDL, RELAX	STEPSP
DIFF1	real	harmonic diffusion constant	ICEMDL	ADVECT, DIFFUS, ICEMDL	
DIFF3	real	harmonic diffusion constant; array with dimensions (*I*MT)	ADVECT	ADVECT, DIFFUS	DIFFU3
DRAGA	real	asymmetric water drag plus the Coriolis parameter; array with dimensions (*I*MT,*J*MT)	FORM	RELAX, ICEMDL	local variable
DRAGS	real	symmetric water drag; array with dimensions (*I*MT,*J*MT)	FORM	RELAX, ICEMDL	local variable
ERROR	real	maximum error allowed in the relation scheme defined by data statement (0.000001)	ICEMDL	ICEMDL, RELAX	local variable
ES	real	NOGAPS atmospheric forcing — surface vapor pressure; array with dimensions (*I*MTP1,*J*MTP1)	RFORCE	HEAT, RFORCE	RFOR
ES1	real	NOGAPS atmospheric forcing — sensible heat flux; array with dimensions (*I*MTP1,*J*MTP1)	RFORCE	HEAT, RFORCE	RFOR
ETA	real	nonlinear shear viscosity; array with dimensions (*I*MT,*J*MT)	PLAST	FORM, ICEMDL, RELAX	local variable
FCORSP	real	array of sine of latitude positions of each gridpoint; array with dimensions (*I*MTP1,*J*MTP1)	DRIVER	DRIVER, FORM, GEOWIND, OCEAN	CORSP
FHEFF	real	total ice growth rate of thick ice: array with dimensions (*I*MT,*J*MT)	HEAT	GROWTH, HEAT, ICEMDL,	GROW
FO	real	growth rate of thin ice; array with dimensions (*N*X1,*N*Y1)	HEAT	GROWTH, HEAT, ICEMDL	local variable
FORCEX	real	x component of forcing due to the ocean currents plus the ice pressure gradient; array with dimensions (*I*MTM1,*J*MTM1)	FORM	RELAX, FORM	FORCE
FORCEY	real	x component of forcing due to the ocean currents plus the ice pressure gradient; array with dimensions (*I*MTM1,*J*MTM1)	FORM	RELAX, FORM	FORCE

NAME	DATA TYPE	DESCRIPTION	CSU DEFINED IN	CSU USED IN	COMMON BLOCK
FSH	real	atmospheric forcing — solar radiation; array with dimensions (*I*MTP1,*J*MTP1)	RFORCE	BUDGET, HEAT, RFORCE	RAD
FW	real	oceanic heat flux; array with dimensions (*I*MT,*J*MT)	RST_OCN	BUDGET, HEAT, OCEAN, RST_OCN, SWAP	OCEANS
FW1	real	heat above the freezing temperature; array with dimensions (*I*MT,*J*MT)	HEAT, RST_OCN	HEAT, RST_OCN, SAV_OCN, TRACER	COX2
FW2	real	oceanic heat flux (watts/m^2); array with dimensions (*I*MT,*J*MT)	BUDGET	BUDGET	COX2
GAIRX	real	x component of the geostrophic wind; array with dimensions (*I*MTP1,*J*MTP1)	RFORCE	DRIVER, FORM, HEAT, ICEMDL, RFORCE, STRESSUP	RFOR2
GAIRY	real	y component of the geostrophic wind; array with dimensions (*I*MTP1,*J*MTP1)	RFORCE	DRIVER, FORM, HEAT, ICEMDL, RFORCE, STRESSUP	RFOR2
GAREA	real	change of compactness due to freezing or melting; array with dimensions (*I*MTP1,*J*MTP1)	GROWTH	ICEMDL	local variable
GICE	real	ice thickness growth rate of open water; array with dimensions (*I*MT,*J*MT)	HEAT, RST_OCN	HEAT, RST_OCN, SAV_OCN, SWAP, TRACER	COX2
GWATX	real	x component of ocean current (m/s); array with dimensions (*I*MTM1,*J*MTM1)	RST_OCN, SWAP	DRIVER, FORM, ICEMDL, RST_OCN, SWAP	RFOR2
GWATY	real	y component of ocean current (m/s); array with dimensions (*I*MTM1,*J*MTM1)	RST_OCN, SWAP	DRIVER, FORM, ICEMDL, RST_OCN, SWAP	RFOR2
HCORR	real	additional ice to be melted for mixed-layer balance; array with dimensions (*I*MT,*J*MT)	GROWTH	ICEMDL	local variable
HDIFF1	real	net rate of total open water growth; array with dimensions (*I*MT,*J*MT)	HEAT	GROWTH, ICEMDL	local variable

NAME	DATA TYPE	DESCRIPTION	CSU DEFINED IN	CSU USED IN	COMMON BLOCK
HEFF	real	mean ice thickness per grid cell; read from restart data file or initialized to constant initial condition; array with dimensions (*I*MT,*J*MT,3)	RST_ICE	ADVECT, DIFFUS, DRIVER, FORM, GROWTH, ICEMDL, HEAT, RNEGT, RST_ICE, RST_OCN, SAV_ICE, SSMI, STRESSUP, XSUM	RSTRT
HEFFM	real	land/sea mask for thermodynamic variables; array with dimensions (*I*MT,*J*MT)	BNDRY	ADVECT, BNDRY, DIFFUS, DRIVER, FORM, GROWTH, ICEMDL, PLAST	MASK
HO	real	demarcation between thick and thin ice defined by data statement; (0.5 m)	ICEMDL	ICEMDL, GROWTH	local variable
IDTG	integer	date-time group; read from standard input (YYMMDDHH)	ICEMDL	DAYNUM, ICEMDL, SWAP	TSTEP
IMT	integer	total number of T grid boxes zonally (360)			local variable
IMTM1	integer	total number of T grid boxes zonally – 1 (359)			local variable
IMTP1	integer	total number of T grid boxes zonally + 1 (361)			local variable
IRSTRT	integer	if 0 restarts from previous run; otherwise, restarts from constant conditions; read from standard input	DRIVER	DRIVER	TSTEP
ITSTEP	integer	number of timesteps for run; read from standard input	DRIVER	DRIVER, ICEMDL	TSTEP
JMT	integer	total number of T grid boxes meridionally (360)			local variable
JMTM1	integer	total number of T grid boxes meridionally – 1 (359)			local variable
JMTP1	integer	total number of T grid boxes meridionally + 1 (361)			local variable
LAD	integer	type of timestepping defined by data statement (2)	ICEMDL	ADVECT, ICEMDL	local variable

NAME	DATA TYPE	DESCRIPTION	CSU DEFINED IN	CSU USED IN	COMMON BLOCK
MDY	integer	day calculated from date-time group	DAYNUM	DAYNUM, DRIVER, ICEMDL, SWAP	TSTEP
MHR	integer	hour calculated from date-time group	DAYNUM	DAYNUM, DRIVER, ICEMDL, SWAP	TSTEP
MIDY	integer	defined by parameter statement (160)	ICEMDL, ADVECT, OCEAN, RELAX	ADVECT, ICEMDL, OCEAN, RELAX	local variable
MM	integer	month calculated from date-time group	DAYNUM	DAYNUM, DRIVER, ICEMDL, SWAP	TSTEP
MYR	integer	year calculated from date-time group	DAYNUM	DAYNUM, DRIVER, ICEMDL, SWAP	TSTEP
NXM1	integer	*I*MT-2; defined by parameter statement (358)	ADVECT	ADVECT	local variable
NYM1	integer	*J*MT-2; defined by parameter statement (358)	ADVECT	ADVECT	local variable
OUT	real	land/sea mask including outflow conditions for thermodynamic variables; array with dimensions (*I*MT,*J*MT)	BNDRY	BNDRY, DRIVER, FORM, GROWTH, ICEMDL	MASK
PLTSTP	integer	interval in timesteps at which to plot results; read from standard input	DRIVER	DRIVER	TSTEP
PRESS	real	ice strength; array with dimensions (*I*MT,*J*MT)	FORM	PLAST, FORM	PRESS
PRTSTP	integer	interval in timesteps at which to write results; read from standard input	DRIVER	DRIVER, ICEMDL	TSTEP
PS	real	NOGAPS atmospheric forcing—surface air pressure; array with dimensions (*I*MTP1,*J*MTP1)	RFORCE	HEAT, GEOWIND, RFORCE	RFOR
PS1	real	NOGAPS atmospheric forcing—total heat flux; array with dimensions (*I*MTP1,*J*MTP1)	RFORCE	HEAT, RFORCE	RFOR
RADIAN	real	deg per radian defined by parameter statement (57.29578°)	ADVECT, DRIVER, OCEAN, RELAX, ICEMDL	ADVECT, DRIVER, OCEAN, RELAX, ICEMDL	local variable

NAME	DATA TYPE	DESCRIPTION	CSU DEFINED IN	CSU USED IN	COMMON BLOCK
RADIUS	real	earth's radius defined by parameter statement (6370.0 * 10^3 m)	ADVECT, CLINIC, ICEMDL, RELAX	ADVECT, CLINIC, ICEMDL, RELAX	local variable
SHICE	real	total ice thickness; array with dimensions (*I*MT,*J*MT)	HEAT, RST_OCN	HEAT, RST_OCN, SAV_OCN, TRACER	COX2
SNRT	real	• Jan, Feb, Mar, or Apr (3.215×10^{-9}) • May (19.29×10^{-9}) • Jun, Jul, or Aug 1–19 (0.0) • Aug 20–31 (49.603×10^{-9}) • Sep or Oct (49.603×10^{-9}) • Nov or Dec (3.215×10^{-9})	ICEMDL	BUDGET, ICEMDL	SNOW
TAIR	real	NOGAPS atmospheric forcing—surface air temperature; array with dimensions (*I*MTP1,*J*MTP1)	RFORCE	BUDGET, HEAT, RFORCE	RFOR
TFRZ	real	temperature at the freezing point; array with dimensions (*I*MT,*J*MT)	RST_OCN, SWAP	HEAT, RST_OCN, SSMI, SWAP	COX2
THETA	real	indicates backward timestep	ICEMDL	ICEMDL, RELAX	local variable
TICE	real	mixed-layer temperature for open water or ice temperature for ice cover (Kelvin); read from restart data file or initialized to constant initial condition; array with dimensions (*I*MT,*J*MT)	RST_ICE	BUDGET, DRIVER, RST_ICE, SAV_ICE	RSTRT
TM1	real	mixed-layer temperature (Kelvin) from previous timestep; array with dimensions (*I*MT,*J*MT)	RST_OCN, SSMI	HEAT, SSMI, RST_OCN, SWAP	COX2
TM2	real	mixed-layer temperature from two timesteps ago (Kelvin); array with dimensions (*I*MT,*J*MT)	RST_OCN, SSMI	RST_OCN, SSMI, SWAP	COX2
TMP	real	ocean salinity and temperature (°C) from previous timestep; array with dimensions (*I*MT,2)	RST_OCN	RST_OCN	COX2
UICE	real	x component of ice drift; array with dimensions (*I*MTM1,*J*MTM1,3)	ICEMDL or RST_ICE	DRIVER, FORM, ICEMDL, PLAST, RELAX, RST_ICE, SAV_ICE, STRESSUP	RSTRT

NAME	DATA TYPE	DESCRIPTION	CSU DEFINED IN	CSU USED IN	COMMON BLOCK
UICEC	real	intermediate x component of the ice drift; read from restart data file or initialized to constant initial condition; array with dimensions (*I*MTM1,*J*MTM1,3)	ICEMDL or RST_ICE	ADVECT, DRIVER, ICEMDL, RELAX, RST_ICE, SAV_ICE	RSTRT
UTEMP	real	x component of ocean current from previous timestep; array with dimension (*I*MT)	RST_OCN	RST_OCN	COX2
UVM	real	land/sea mask for velocity variables; array with dimensions (*I*MTM1,*J*MTM1)	BNDRY	BNDRY, ICEMDL, RELAX	MASK
VICE	real	y component of ice drift; array with dimensions (*I*MTM1,*J*MTM1,3)	ICEMDL or RST_ICE	DRIVER, FORM, ICEMDL, PLAST, RELAX, RST_ICE, SAV_ICE, STRESSUP	RSTRT
VICEC	real	intermediate y component of the ice drift; read from restart data file or initialized to constant initial condition; array with dimensions (*I*MTM1,*J*MTM1,3)	ICEMDL or RST_ICE	ADVECT, DRIVER, ICEMDL, RELAX, RST_ICE, SAV_ICE	RSTRT
VTEMP	real	y component of ocean current from previous timestep; array with dimension (*I*MT)	RST_OCN	RST_OCN	COX2
ZETA	real	nonlinear bulk viscosity; array with dimensions (*I*MT,*J*MT)	PLAST	FORM, ICEMDL, RELAX	local variable

4.2 Ocean Data Table

This section describes the global data elements of CSC Compute Ocean.

DATA ELEMENT	DATA TYPE	DESCRIPTION	CSU DEFINED IN	CSU USED IN	COMMON BLOCK
ABT	real	temporary array for printing topography map	OCEAN	OCEAN	local variable
ACOR	real	if = 0, treat the Coriolis term explicitly if > 0, treat the Coriolis term implicitly with forward component weighted by ACOR, past component by 1 – ACOR	OCEAN	CLINIC, RELAX	SCALAR
ACORF	real	= ACOR; read in NAMELIST PARMS	OCEAN	OCEAN	local variable
AH	real	coefficient of horizontal mixing of T	OCEAN	STEP	SCALAR
AHF	real	= AH; read in NAMELIST EDDY	OCEAN	OCEAN	local variable
AICE	real	ice concentration read from restart file created by the ice model	OCEAN	TRACER	TSTOP
AKNTRL	real	available for storage of values that need to be saved to the restart file	not used	not used	FULLWD
AM	real	coefficient of horizontal mixing of U,V	OCEAN	CLINIC	SCALAR
AMF	real	= AM; read in NAMELIST EDDY	OCEAN	OCEAN	local variable
AREA	real	area of the surface of the model basin	OCEAN		FULLWD
ASH	real	set to 0 for a grid cell that meets the criteria for open water	TRACER	TRACER	local variable
BBTJ	real	coefficient used in horizontal mixing of T	TRACER	TRACER	local variable
BBUJ	real	coefficient used in horizontal mixing of U,V	CLINIC	CLINIC	local variable
BCON	real	slab incidental data on $N+1$ slab	STEP	STEP	WORKSP
BIG	real	virtual disk, stores disk data in core-contained mode	ODAM	ODAM	BG

DATA ELEMENT	DATA TYPE	DESCRIPTION	CSU DEFINED IN	CSU USED IN	COMMON BLOCK
BLK	character*1	a single blank character	OCEAN	OCEAN	local variable
BOXVOL	real	volume of one grid box	TRACER	TRACER	local variable
BUOY	real	energy transfer through buoyancy effects	TRACER, STEP	TRACER	FULLWD
C	real	coefficients of equation of state	STATE	STATE	local variable
C2DTSF	real	DTSF × 2	STEP	CLINIC	SCALAR
C2DTTS	real	DTTS × 2	STEP	TRACER	SCALAR
C2DTUV	real	DTUV × 2	STEP	CLINIC	SCALAR
C2DZ	real	DZ × 2	OCEAN	STEP	ONEDIM
C2DZQ	real	DZQ × 2	STEP	CLINIC, TRACER	WORKSP
CCTJ	real	coefficient used in horizontal mixing of T	STEP TRACER	STEP, TRACER	local variable
CCUJ	real	coefficient used in horizontal mixing of U,V	CLINIC	CLINIC	local variable
CD	real	a constant used in CSU CLINIC to compute vertical diffusion of momentum; = 0.0013	CLINIC	CLINIC	local variable
CFE	real	coefficient of eastern point in LaPlacian star	RELAX	RELAX	WORKSP (R)
CFN	real	coefficient of northern point in LaPlacian star	RELAX	RELAX	WORKSP (R)
CFS	real	coefficient of southern point in LaPlacian star	RELAX	RELAX	WORKSP (R)
CFW	real	coefficient of western point in LaPlacian star	RELAX	RELAX	WORKSP (R)
CHENG	real	Levitus climatological data of ocean temperature and salinity	OCEAN	TRACER	TSTOP
CIQ	real	Knudsen formula coefficients with alternating reference levels	STATE	STATE	WORKSP
COF	real	normalization array in computation of island flow	RELAX	RELAX	WORKSP (R)

DATA ELEMENT	DATA TYPE	DESCRIPTION	CSU DEFINED IN	CSU USED IN	COMMON BLOCK
COFIS	real	integral of COF	RELAX	RELAX	WORKSP (R)
CONTRL	real	NAMELIST data file of program flow controllers	OCEAN	OCEAN	local variable
CPF	real	normalization factor used in constructing LaPlacian star	RELAX	RELAX	WORKSP (R)
CQ	real	Knudsen formula coefficients	STATE	STATE	WORKSP
CRIT	real	criterion for convergence of relaxation	OCEAN	RELAX	SCALAR
CRITF	real	= CRIT; read in NAMELIST PARMS	OCEAN	OCEAN	local variable
CRTP	real	altered value of CRIT for computational efficiency	RELAX	RELAX	local variable
CS	real	cosine of U,V point latitudes	OCEAN	CLNIIC, TRACER	ONEDIM
CSR	real	1.0/CS	OCEAN	CLINIC	ONEDIM
CST	real	cosine of T point latitudes	OCEAN	CLINIC, TRACER	ONEDIM
CSTR	real	1.0/CST	OCEAN	CLINIC, TRACER	ONEDIM
DAYSYR	real	number of days per year; used in CSU STEP to print timestep information.	STEP	STEP	local variable
DDTJ	real	coefficient used in horizontal mixing of T	TRACER	TRACER	local variable
DDUJ	real	coefficient used in horizontal mixing of U,V	CLINIC	CLINIC	local variable
DETMR	real	reciprocal of determinant of the matrix arising from simultaneous equations of the semi-implicit treatment of the Coriolis term	CLINIC	CLINIC	local variable
DIAG1	real	temporary storage of diagonal differences for computational efficiency	STEP	STEP	local variable
DIAG2	character*1	temporary storage of diagonal differences for computational efficiency	STEP	STEP	local variable

DATA ELEMENTS	DATA TYPE	DESCRIPTION	CSU DEFINED IN	CSU USED IN	COMMON BLOCK
DOT	real	= '.'	OCEAN	OCEAN	local variable
DPDX	real	zonal derivative of hydrostatic pressure; used in computing zonal component of internal mode velocity	CLINIC	CLINIC	local variable
DPDY	real	meridional derivative of hydrostatic pressure; used in computing meridional component of internal mode velocity	CLINIC	CLINIC	local variable
DTABS	real	volume average of absolute change of temperature	TRACER, STEP	STEP	FULLWD
DTSF	real	length of timestep on stream function	OCEAN	STEP	SCALAR
DTSFF	real	= DTSF; read in NAMELIST TSTEP	OCEAN	OCEAN	local variable
DTTS	real	length of timestep on T	OCEAN	STEP	SCALAR
DTTSF	real	= DTTS; read in NAMELIST TSTEP	OCEAN	OCEAN	local variable
DTUV	real	length of timestep on U,V	OCEAN	STEP	SCALAR
DTUVF	real	= DTUV; read in NAMELIST TSTEP	OCEAN	OCEAN	local variable
DXT	real	zonal grid spacing across T boxes (between U,V points)	OCEAN	STEP, TRACER	ONEDIM
DXT2R	real	1.0/(DXT * 2.0)	OCEAN	CLINIC	ONEDIM
DXT4R	real	1.0/(DXT * 4.0)	OCEAN	TRACER	ONEDIM
DXT4RQ	real	similar to DXT4R but for vectorization	STEP	CLINIC, TRACER	WORKSP
DXTQ	real	similar to DXT but for vectorization	STEP	TRACER	WORKSP
DXTR	real	1.0/DXT	OCEAN	RELAX, CLINIC	ONEDIM
DXU	real	zonal grid spacing across U,V boxes (between T points)	OCEAN	CLINIC	ONEDIM
DXU2R	real	1.0/(DXU * 2)	OCEAN	CLINIC, TRACER	ONEDIM
DXU2RQ	real	similar to DXU2R but for vectorization	STEP	CLINIC	WORKSP

DATA ELEMENT	DATA TYPE	DESCRIPTION	CSU DEFINED IN	CSU USED IN	COMMON BLOCK
DXU4R	real	1.0/(DXU * 4)	OCEAN	not used	ONEDIM
DXUQ	real	similar to DXU but for vectorization	STEP	TRACER	WORKSP
DXUR	real	1.0/DXU	OCEAN	CLINIC	ONEDIM
DYT	real	meridional grid spacing across T boxes (between U,V points)	OCEAN	TRACER	ONEDIM
DYT2R	real	1.0/(DYT * 2.0)	OCEAN	CLINIC	ONEDIM
DYT4R	real	1.0/(DYT * 4.0)	OCEAN	not used	ONEDIM
DYTR	real	1.0/DYT	OCEAN	CLINIC, TRACER, RELAX	ONEDIM
DYU	real	meridional grid spacing across U,V boxes (between T points)	OCEAN	CLINIC, TRACER	ONEDIM
DYU2R	real	1.0/(DYU * 2.0)	OCEAN	CLINIC	ONEDIM
DYU4R	real	1.0/(DYU * 4.0)	OCEAN	CLINIC	ONEDIM
DYUR	real	1.0/DYU	OCEAN	CLINIC	ONEDIM
DZ	real	vertical grid spacing down U,V,T boxes (between W points); vertical layer thickness in centimeters	OCEAN	CLINIC, TRACER	ONEDIM
DZ2R	real	1.0/(DZU * 2.0)	OCEAN	STEP, TRACER	ONEDIM
DZ2RQ	real	similar to DZ2R but for vectorization	STEP	TRACER	WORKSP
DZZ	real	vertical grid spacing across W boxes (between U,V,T points)	OCEAN	OCEAN, CLINIC	ONEDIM
DZZ2R	real	1.0/(DZZ * 2.0)	OCEAN	TRACER	ONEDIM
DZZ2RQ	real	similar to DZZ2R but for vectorization	STEP	not used	WORKSP
DZZQ	logical	similar to DZZ but for vectorization	STEP	CLINIC	WORKSP
EB	namelist	if true, Euler backward step for time mixing; if false, forward timestep for time mixing	OCEAN	RELAX, STEP	FULLWD

DATA ELEMENT	DATA TYPE	DESCRIPTION	CSU DEFINED IN	CSU USED IN	COMMON BLOCK
EDDY	real	NAMELIST data file of eddy mixing coefficients	OCEAN	OCEAN	local variable
EEH	real	upper vertical mixing coefficient of T	OCEAN	TRACER	ONEDIM
EEHQ	real	similar to EEH but for vectorization	STEP	TRACER	WORKSP
EEM	real	upper vertical mixing coefficient of U,V	OCEAN	CLINIC	ONEDIM
EEMQ	real	similar to EEM but for vectorization	STEP	CLINIC	WORKSP
EKTOT	real	total kinetic energy normalized by volume	STEP, CLINIC	STEP, CLINIC	FULLWD
ENGEXT	real	accumulators of rates of change of kinetic energy of external mode	STEP, CLINIC	STEP, CLINIC	FULLWD
ENGINT	real	accumulators of rates of change of kinetic energy of internal mode	STEP, CLINIC	STEP, CLINIC	FULLWD
ENGTMP	real	temporary accumulator of change of kinetic energy of external mode	CLINIC	CLINIC	local variable
FFH	real	lower vertical mixing coefficient of T	OCEAN	TRACER	ONEDIM
FFHQ	real	similar to FFH but for vectorization	STEP	TRACER	WORKSP
FFM	real	lower vertical mixing coefficient of U,V	OCEAN	CLINIC	ONEDIM
FFMQ	real	similar to FFM but for vectorization	STEP	CLINIC	WORKSP
FINS	real	floating point array to read in start and end indices	OCEAN	OCEAN	local variable
FKMP	real	number of vertical levels of ocean at T points	OCEAN	OCEAN	local variable
FKMQ	real	number of vertical levels of ocean at U,V points	OCEAN	OCEAN	local variable
FKMT	real	number of vertical levels of ocean at T points	OCEAN	STEP	WORKSP
FKMTM	real	FKMT at row $J-1$	not used	not used	WORKSP
FKMTP	real	FKMT at row $J+1$	STEP	STEP	WORKSP

DATA ELEMENT	DATA TYPE	DESCRIPTION	CSU DEFINED IN	CSU USED IN	COMMON BLOCK
FKMU	real	number of vertical levels of ocean at U,V points	OCEAN	STEP	WORKSP
FKMUM	real	FKMU at row $J-1$	not used	not used	WORKSP
FKMUP	real	FKMU at row $J+1$	STEP	STEP	WORKSP
FKMZ	real	number of vertical levels of ocean at interior (non-land neighboring) T points	OCEAN	OCEAN	local variable
FKPH	real	coefficient of vertical mixing of T	OCEAN	OCEAN	SCALAR
FKPHF	real	= FKPH; read in NAMELIST EDDY	OCEAN	OCEAN	local variable
FKPM	real	coefficient of vertical mixing of U,V	OCEAN	OCEAN	SCALAR
FKPMF	real	= FKPM; read in NAMELIST EDDY	OCEAN	OCEAN	local
FM	real	masking array for T points; 0 over land and 1 over ocean	STEP	STEP, TRACER	WORKSP
FMM	real	FM at row $J-1$	STEP	STEP	WORKSP
FMP	real	FM at row $J+1$	STEP	STEP, TRACER	WORKSP
FUW	real	zonal component of advective coefficient for west face of U,V box in CLINIC and of T box in TRACER	CLINIC, TRACER	CLINIC, TRACER	WORKSP
FVN	real	meridional component of advective coefficient for north face of U,V box in CLINIC and of T box in TRACER	CLINIC, TRACER	CLINIC, TRACER	WORKSP
FVST	real	advective coefficient for south face of T box	TRACER	TRACER	WORKSP
FVSU	real	advective coefficient for south face of U,V box	STEP	CLINIC	WORKSP
FX	real	temporary value, constant in subsequent DO loops	CLINIC, TRACER, STEP, RELAX	CLINIC, TRACER, STEP, RELAX	local variable
FXA	real	temporary value, constant in subsequent DO loops	CLINIC, TRACER, RELAX	CLINIC, TRACER, RELAX	local variable

DATA ELEMENT	DATA TYPE	DESCRIPTION	CSU DEFINED IN	CSU USED IN	COMMON BLOCK
FXB	real	temporary value, constant in subsequent DO loops	CLINIC, TRACER	CLINIC, TRACER	local variable
FXC	real	temporary value, constant in subsequent DO loops	RELAX	RELAX	local variable
FW	real	not used; deep oceanic heat fluxes read in from the ice model	not used	not used	TSTOP
GGUJ	real	coefficient used in horizontal mixing of U,V	CLINIC	CLINIC	local variable
GICE	real	ice growth rates computed by the ice model	OCEAN	TRACER	TSTOP
GM	real	masking array for U,V points; 0 for land and 1 for ocean; used in equations to force 0 values for land grid cells	STEP	CLINIC	WORKSP
GRAV	real	acceleration due to gravity = 980.6 cm/s^2	OCEAN	CLINIC, TRACER	SCALAR
GRID	real	0.5715; defined in PARAMETER statement	all	OCEAN	local variable
GRID2	real	0.28575; defined in PARAMETER statement	all	OCEAN	local variable
GRID3	real	0.28575; defined in PARAMETER statement	all	OCEAN	local variable
HHUJ	real	coefficient used in horizontal mixing of U,V	CLINIC	CLINIC	local variable
HICE	real	ice thickness computed from the ice model	OCEAN	TRACER	TSTOP
HR	integer	reciprocal of total depth at U,V points	OCEAN	CLINIC, STEP	FIELDS
I	integer	zonal gridpoint index	all	all	local variable
IBK	integer	DO loop index	OCEAN	OCEAN	local variable
IBOX		NAMELIST data file of island box corner point indices	OCEAN	OCEAN	local variable
IE	integer	ending index for *I* DO loop	CLINIC, RELAX, MATRIX	CLINIC, RELAX, MATRIX	local variable

DATA ELEMENT	DATA TYPE	DESCRIPTION	CSU DEFINED IN	CSU USED IN	COMMON BLOCK
IEIS	integer	ending index for island box	OCEAN	RELAX	FULLWD
IEPT	integer	index designator	OCEAN	OCEAN	local variable
IEPU	integer	index designator	OCEAN	OCEAN	local variable
IEZ	integer	array of ending indices for vorticity	OCEAN	RELAX	FULLWD
IFKMP	integer	number of vertical levels of ocean at T points read from bathymetry file if starting a run from scratch	OCEAN	OCEAN	local variable
ILA	integer	DO loop index	OCEAN	OCEAN	local variable
ILO	integer	DO loop index	OCEAN	OCEAN	local variable
IMT	integer	total number of T grid boxes zonally; = 360; defined in PARAMETER statement	all	all	local variable
IMTKM	integer	*I*MT * *K*M; = 1500; defined in PARAMETER statement	all	all	local variable
IMTM1	integer	*I*MT – 1; = 359; defined in PARAMETER statement	all	all	local variable
IMTM2	integer	*I*MT – 2; = 358; defined in PARAMETER statement	all	not used	local variable
IMTP1	integer	*I*MT + 1; = 361; defined in PARAMETER statement	all	all	local variable
IMU	integer	total number of U,V grid boxes zonally; = 359; defined in PARAMETER statement	all	all	local variable
IMUM1	integer	*I*MU – 1; = 358; defined in PARAMETER statement	all	CLINIC, OCEAN, STEP	local variable
IMUM2	integer	*I*MU – 2; = 357; defined in PARAMETER statement	all	not used	local variable
IND	integer	indicates which levels to compare for static stability	STATE	STATE	local variable
INT2	integer	1; defined in PARAMETER statement	all	all	local variable
INT3	integer	1; defined in PARAMETER statement	all	all	local variable

DATA ELEMENT	DATA TYPE	DESCRIPTION	CSU DEFINED IN	CSU USED IN	COMMON BLOCK
INTVL2	integer	2; defined in PARAMETER statement	all	all	local variable
INTVL3	integer	2; defined in PARAMETER statement	all	all	local variable
IPRT	integer	number of columns to print beginning at western boundary	STEP	STEP	local variable
IS	integer	starting index for *I* DO loop	RELAX	RELAX	local variable
ISIS	integer	starting *I* index of island box	OCEAN	OCEAN	FULLWD
ISLE	integer	DO loop index indicating island being computed	RELAX	RELAX	local variable
ISMASK	integer	gridpoint type indicator: = 0 over interior points = 1 over perimeter points = 2 over land points	RELAX	RELAX	WORKSP (R)
ISP	integer	index designator	OCEAN	OCEAN	local variable
ISTEP	integer	number of timesteps in a day	STEP	STEP	local variable
ISTOP	integer	final column to be printed	STEP	STEP	local variable
ISTRT	integer	first column to be printed	STEP	STEP, MATRIX	local variable
ISZ	integer	array of starting indices for vorticity	OCEAN	RELAX	FULLWD
ITT	integer	timestep counter; total number of timesteps completed	OCEAN	STEP, CLINIC, TRACER, STRESSUP, RELAX, OCEAN	FULLWD
J	integer	meridional gridpoint index	all	all	local variable
JE	integer	ending index for *J* DO loop	RELAX	RELAX	local variable
JEIS	integer	ending *J* index of island box	OCEAN	RELAX	FULLWD
JJ	integer	index designator	STEP	STEP	local variable

DATA ELEMENT	DATA TYPE	DESCRIPTION	CSU DEFINED IN	CSU USED IN	COMMON BLOCK
JMT	integer	total number of T grid boxes meridionally; = 360; defined in PARAMETER statement	all	all	local variable
JMTM1	integer	JMT – 1; = 359; defined in PARAMETER statement	all	OCEAN, RELAX, STEP, STRESSUP	local variable
JMTM2	integer	JMT – 2; = 358; defined in PARAMETER statement	all	STEP	local variable
JMTP1	integer	JMT + 1; = 361; defined in PARAMETER statement	all	all	local variable
JREV	integer	DO loop index in reverse order	OCEAN	OCEAN	local variable
JS	integer	starting index for *J* DO loop	RELAX	RELAX	local variable
JSCAN	integer	= JMTM2; defined in PARAMETER statement	all	OCEAN, RELAX	local variable
JSIS	integer	starting *J* index of island box	OCEAN	RELAX	FULLWD
K	integer	vertical gridpoint index	all	all	local variable
KAR	integer	KAR(K) = K; used to enable vectorization	OCEAN	STEP, CLINIC	FULLWD
KFLDS	integer	disk unit number (12) for two-dimensional horizontal fields and start and end indices	OCEAN	OCEAN, RELAX, STEP	FULLWD
KM	integer	total number of vertical levels; = 15; defined in PARAMETER statement	all	all	local variable
KMM1	integer	*K*M – 1; = 14; defined in PARAMETER statement	all	CLINIC, TRACER	local variable
KMP1	integer	*K*M + 1; = 16; defined in PARAMETER statement	all	all	local variable
KMP2	integer	*K*M + 2; = 17; defined in PARAMETER statement	all	OCEAN,STEP, TRACER, CLINIC, STATE	local variable
KMT	integer	number of vertical levels of ocean at T points	STEP	STEP, TRACER	FULLWD
KMTP	integer	*K*MT + 1	STEP	STEP	FULLWD
KMU	integer	number of vertical levels of ocean at U,V points	STEP	STEP, CLINIC	FULLWD

DATA ELEMENT	DATA TYPE	DESCRIPTION	CSU DEFINED IN	CSU USED IN	COMMON BLOCK
KMUP	integer	*K*MU + 1	STEP	CLINIC	FULLWD
KONTRL	integer	disk unit number for timestep counter, etc.	OCEAN	OCEAN, STEP	FULLWD
KPR	integer	temporary array for printing topography map	OCEAN	OCEAN	local variable
KREF	integer	reference level indicator	STATE	STATE	local variable
KS	integer	indicates which levels to compare for static stability	TRACER	TRACER	local variable
KZ	integer	temporary indicator of number of levels of ocean points	OCEAN, CLINIC, TRACER	OCEAN, CLINIC, TRACER	local variable
L	integer	index designator	OCEAN, MATRIX	OCEAN, MATRIX	local variable
LABS	integer	disk unit numbers (13–15) for slabs	OCEAN	OCEAN, STEP	FULLWD
LBC	integer	number of arrays of slab incidental data; = 2; defined in PARAMETER statement	all	all	local variable
LL	integer	index designator	STEP	STEP, CLINIC	local variable
LO	integer	restart tape unit number	ODAM	ODAM	local variable
LSEG	integer	maximum number of sets of start and end indices; = 10 for vorticity; defined in PARAMETER statement	all	all	local variable
LSEGP	integer	LSEG + 1	OCEAN	OCEAN	local variable
LU	integer	disk unit number	ODAM	ODAM	local variable
LUPTD	integer	permuting disk unit number for saving previous relaxation solution	RELAX	RELAX	local variable
LUPTDB	integer	permuting disk unit number for saving relaxation solution of two timesteps previous; if LUPTD = 5 then LUPTDB = 6; when next solution is obtained, LUPTDB is set to 5, LUPTD to 6 and the new solution is written to LUPTD	RELAX	RELAX	local variable

DATA ELEMENT	DATA TYPE	DESCRIPTION	CSU DEFINED IN	CSU USED IN	COMMON BLOCK
M	integer	index indicating the tracer being computed	TRACER	TRACER	local variable
MIDX	integer	100; defined in PARAMETER statement	all	all	local variable
MIDXX	integer	200; defined in PARAMETER statement	all	none	local variable
MIDY	integer	80; defined in PARAMETER statement	all	all	local variable
MIDYY	integer	160; defined in PARAMETER statement	all	none	local variable
MIX	integer	mixing timestep indicator; if 0, not a mixing timestep; if = 1, mixing timestep	STEP	RELAX, STEP	FULLWD
MSB	integer	not used at present	not used	not used	FULLWD
MSCAN	integer	relaxation scan counter	RELAX	RELAX	FULLWD
MTEST	integer	print flag	STEP	STEP	local variable
MXP	integer	if = 1, second pass of Euler backward timestep	STEP	STEP, CLINIC, TRACER, RELAX	FULLWD
MXSCAN	integer	maximum number of relaxation scans permitted	OCEAN	RELAX	FULLWD
N	integer	index designator	OCEAN, STEP, STATE, ODAM	OCEAN, STEP, STATE, ODAM	local variable
NA	integer	if = 1, write a restart file	OCEAN	OCEAN	FULLWD
NB	integer	read in by NAMELIST CONTRL	OCEAN	not used	FULLWD
NBLK	integer	number of words per block on disk unit; argument in CSU ODAM	not used	not used	local variable
NBUF	integer	number of buffers set aside for disk I/O use; only used when not running in core-contained mode	OCEAN	not used	local variable
NC	integer	read in by NAMELIST CONTRL	OCEAN	not used	FULLWD

DATA ELEMENT	DATA TYPE	DESCRIPTION	CSU DEFINED IN	CSU USED IN	COMMON BLOCK
NCON	integer	number of passes to make on convection; since mixing is done only two levels at a time, total homogenization of unstable levels does not occur in one pass	TRACER	TRACER	local variable
NDICES	integer	total number of start and end indices; = 7216; defined in PARAMETER statement	all	all	local variable
NDISK	integer	permutes with NDISKB and NDISKA on 13, 14, 15 indicating the disk units for the slabs at various time levels	OCEAN, STEP	OCEAN, STEP	FULLWD
NDISKA	integer	see NDISK	OCEAN, STEP	OCEAN, STEP	FULLWD
NDISKB	integer	see NDISK	STEP	STEP	FULLWD
NDISKX	integer	temporary disk unit to which data is written on second pass of Euler backward timestep	STEP	STEP	local variable
NDW	integer	unused integer read in by NAMELIST CONTRL; may be used to indicate writeout of data for analysis purposes	OCEAN	not used	FULLWD
NE	integer	ending DO loop limit	ODAM	ODAM	local variable
NERGY	integer	if = 1, indicates energy printout timestep; if = 0, not an energy printout timestep	STEP	STEP, CLINIC, TRACER	FULLWD
NFIRST	integer	if = 1, start a run from scratch if = 0, restart from data supplied on units 11–15; read in by NAMELIST CONTRL	OCEAN	OCEAN	FULLWD
NFRST	integer	disk unit address of the first word to be transferred	ODAM	ODAM	local variable
NGRDP	integer	= 181; defined in PARAMETER statement	all	all	local variable
NGRDPX	integer	= 361; defined in PARAMETER statement	all	all	local variable

DATA ELEMENT	DATA TYPE	DESCRIPTION	CSU DEFINED IN	CSU USED IN	COMMON BLOCK
NGRDPY	integer	= 361; defined in PARAMETER statement	all	all	local variable
NIEVEN	integer	4; defined in PARAMETER statement	all	none	local variable
NISLE	integer	number of islands in the model basin; = 4; defined in PARAMETER statement	all	all	local variable
NKFLDS	integer	number of two-dimensional fields needed on disk unit 12; this is six plus the number of (equivalent) two-dimensional fields needed to contain the start and end indices, normally one; = 7; defined in PARAMETER statement	all	OCEAN, ODAM, STRESSUP	local variable
NLAST	integer	final timestep to compute on this run of the model; read in by NAMELIST CONTRL	OCEAN	OCEAN, STEP	FULLWD
NMIX	integer	number of timesteps between mixing timesteps; mixing is done to suppress the computational mode associated with leap-frog timestepping	OCEAN	STEP	FULLWD
NNERGY	integer	number of timesteps between execution of energy/printout code; read in by NAMELIST CONTRL	OCEAN	STEP	FULLWD
NRTLF1	integer	178; defined in PARAMETER statement	all	all	local variable
NS	integer	starting DO loop limit	ODAM, STATE	ODAM, STATE	local variable
NSKP	integer	index designator	ODAM	ODAM	local variable
NSLAB	integer	number of words in one slab; = 22320; defined in PARAMETER statement	all	OCEAN, STATE, STRESSUP	local variable
NSTLF1	integer	0; defined in PARAMETER statement	all	all	local variable
NSTLW1	integer	0; defined in PARAMETER statement	all	all	local variable

DATA ELEMENT	DATA TYPE	DESCRIPTION	CSU DEFINED IN	CSU USED IN	COMMON BLOCK
NSTRT1	integer	180; defined in PARAMETER statement	all	all	local variable
NSTUP1	integer	180; defined in PARAMETER statement	all	all	local variable
NSWICH	integer	number of words in slab incidental data that are nonprognostic and must be switched; = 720; defined in PARAMETER statement	all	STEP	local variable
NT	integer	number of tracer type variables carried in the model; temperature and salinity (2) + number of passive tracers (0); = 2; defined in PARAMETER statement	all	all	local variable
NTMIN2	integer	maximum of NT or 2; = 2; defined in PARAMETER statement	all	all	local variable
NTOT	integer	total length of a disk unit; argument in CSU ODAM	not used	not used	local variable
NTSI	integer	number of timesteps between print of a single line of information containing timestep number, kinetic energy, etc.	OCEAN	STEP, CLINIC, TRACER	FULLWD
NUPLW1	integer	178; defined in PARAMETER statement	all	all	local variable
NWDS	integer	*IMT*JMT*; in ODAM: number of words to transfer in a disk operation = 129600; defined in PARAMETER statement	all	all	local variable
NWRITE	integer	used in core-contained mode as number of timesteps between backup restart write	OCEAN	OCEAN	FULLWD
NWRS	real	number of words to transfer in a disk operation	ODAM	ODAM	local variable
OMEGA	real	rate of rotation of the coordinate system	OCEAN	CLINIC	SCALAR
P	real	mass transport stream function	RELAX	CLINIC, STEP	FIELDS

DATA ELEMENT	DATA TYPE	DESCRIPTION	CSU DEFINED IN	CSU USED IN	COMMON BLOCK
PAD	namelist	unused memory space to prevent overwriting on I/O	not used	not used	FULLWD
PARMS	real	NAMELIST data file of various parameters in the model	OCEAN	OCEAN	FULLWD
PB	real	P in previous timestep	STEP, OCEAN	CLINIC, RELAX, STEP	FIELDS
PHI	real	latitude in radians of the U,V points	OCEAN	OCEAN	ONEDIM
PHIT	real	latitude in radians of the T points	OCEAN	OCEAN	ONEDIM
PI	real	pi; = 3.1415927	OCEAN	not used	local variable
PLICEX	real	energy change due to implicit effects on external mode	STEP	STEP	FULLWD
PLICIN	real	energy change due to implicit effects on internal mode	STEP	STEP	FULLWD
PTD	real	change of stream function across a timestep	RELAX	RELAX	local variable
PTDB	real	change of stream function across previous timestep	RELAX	RELAX	local variable
RADIAN	real	radian to degree conversion factor; = 57.29578	OCEAN	OCEAN	SCALAR
RADIUS	real	radius of the Earth in centimeters; 6730.E5	OCEAN	CLINIC	SCALAR
RES	real	residual of relaxation	RELAX	RELAX	WORKSP (R)
RESIS	real	residual of relaxation of island	RELAX	RELAX	local variable
RESMAX	real	maximum of the residuals at all gridpoints	RELAX	RELAX	local variable
RHO	real	density with a space and time invariant constant subtracted	STATE	STATE	local variable
RHON	real	RHO for the row to the north	CLINIC	TRACER	WORKSP
RHOS	real	RHO for the row to the south	STEP	CLINIC, TRACER	WORKSP

DATA ELEMENT	DATA TYPE	DESCRIPTION	CSU DEFINED IN	CSU USED IN	COMMON BLOCK
RICE0	real	ratio of the latent heat of fusion of sea ice to the heat capacity of water	TRACER	TRACER	local variable
SCL	real	scaling factor for printout	STEP	STEP	local variable
SFU	real	external mode component of U; also used for temporary storage	STEP, CLINIC	STEP, CLINIC	ONEDIM
SFUB	real	SFU in previous timestep	STEP, CLINIC	STEP, CLINIC	ONEDIM
SFV	real	external mode component of V; also used for temporary storage	STEP, CLINIC	STEP, CLINIC	ONEDIM
SFVB	real	SFV in previous timestep	STEP, CLINIC	STEP, CLINIC	ONEDIM
SHICE	real	growth rate of ice thickness	OCEAN	TRACER	TSTOP
SINE	real	sine of U,V point latitude	OCEAN	CLINIC	ONEDIM
SINEA	real	sine of U,V point latitude in the Earth-oriented coordinates used to compute the Coriolis force	OCEAN	STEP, CLINIC	ONEDIM
SMIX	real	interpolated monthly salinity data from the Levitus climatology	OCEAN	TRACER, OCEAN	TSTOP
SO	real	normalizing salinities for Knudsen coefficients	STATE	STATE	local variable
SOIQ	real	normalizing salinities with alternating reference level	STATE	STATE	WORKSP
SOQ	real	same as SO but for vectorization	STATE	STATE	WORKSP
SOR	real	coefficient of over-relaxation; normally between 1.5–1.8	OCEAN	RELAX	SCALAR
SORF	real	= SOR; read in NAMELIST PARMS	OCEAN	OCEAN	local variable
SQ	real	normalized salinities	STATE	STATE	local variable
STLF1	real	0; defined in PARAMETER statement	all	all	local variable
STLW1	real	0; defined in PARAMETER statement	all	all	local variable

DATA ELEMENT	DATA TYPE	DESCRIPTION	CSU DEFINED IN	CSU USED IN	COMMON BLOCK
STRT1	real	180; defined in PARAMETER statement	all	all	local variable
STUP1	real	180; defined in PARAMETER statement	all	all	local variable
SUMDY	real	summation of DYT	OCEAN	OCEAN	local variable
SWLDEG	real	= −45.72; latitude in degrees of the southern wall	OCEAN	OCEAN	SCALAR
SX	real	salinities; dummy argument in CSU STATE	STATE	STATE	local variable
T	real	tracer type of variables (temperature, salinity)	OCEAN, TRACER	STEP, TRACER	WORKSP
TA	real	similar to T but after present	TRACER	TRACER	WORKSP
TB	real	similar to T but before present	OCEAN	STEP, TRACER	WORKSP
TBM	real	similar to TB but in row $J-1$	STEP	STEP, TRACER	WORKSP
TBP	real	similar to TB but in row $J+1$	STEP	STEP, TRACER	WORKSP
TBRN	real	zonal summation of tracer defined to the north	STEP	STEP	local variable
TBRS	real	zonal summation of tracer defined to the south	STEP	STEP	local variable
TBRZ	real	zonal/vertical average of tracer	STEP	STEP	local variable
TBSLAB	real	array of entire N–1 slab; equivalenced to TB	STEP	STEP	local variable
TCHENG	real	interpolated monthly Levitus data of temperature and salinity	TRACER	TRACER	TSTOP
TDIF	real	diffusion computation array	TRACER	CLINIC, TRACER, STEP	WORKSP
TEMPA	real	utility array used as temporary storage	CLINIC, TRACER	CLINIC, TRACER	WORKSP
TEMPB	real	utility array used as temporary storage	CLINIC, TRACER	CLINIC, TRACER	WORKSP
TEST1	real	product of the reciprocals of four surrounding depths	RELAX	RELAX	local variable

DATA ELEMENT	DATA TYPE	DESCRIPTION	CSU DEFINED IN	CSU USED IN	COMMON BLOCK
TEST2	real	sum of the reciprocals of four surrounding depths	RELAX	RELAX	local variable
TICE	real	ice surface temperature computed by the ice model	OCEAN	not used	TSTOP
TINIT	real	initial values of tracers for starting from scratch	OCEAN	OCEAN	ONEDIM
TINITF	real	= TINIT; read from NAMELIST TSPROF	OCEAN	OCEAN	local variable
TM	real	similar to T but in row $J-1$	STEP	TRACER	WORKSP
TMIX	real	interpolated monthly temperature data from the Levitus climatology	OCEAN	OCEAN	TSTOP
TMPCHG	real	temporary storage	TRACER	TRACER	TSTOP
TMT	real	meridional mass transport	STEP	STEP	local variable
TNG	real	tangent of U,V point latitude	OCEAN	CLINIC	ONEDIM
TO	real	normalizing temperature for Knudsen coefficients	STATE	STATE	local variable
TOIQ	real	normalizing temperature with alternating reference levels	STATE	STATE	WORKSP
TOQ	real	similar to TO but for vectorization	STATE	STATE	WORKSP
TOTDX	real	total zonal span of ocean boxes	STEP	STEP	local variable
TOTDZ	real	total vertical span of ocean boxes	STEP	STEP	local variable
TP	real	similar to T but in row $J+1$	STEP	STEP, TRACER, CLINIC	WORKSP
TQ	real	normalized temperature	STATE	STATE	local variable
TSLAB	real	array of entire N slab; equivalenced to T in CSU STEP	STEP	STEP	local variable
TSPROF	namelist	NAMELIST data file for initial tracers	OCEAN	OCEAN	local variable
TSTEPS	namelist	NAMELIST data file for timestep lengths	OCEAN	OCEAN	local variable

DATA ELEMENT	DATA TYPE	DESCRIPTION	CSU DEFINED IN	CSU USED IN	COMMON BLOCK
TTDAY	real	current day of the year number	STEP	STEP	local variable
TTDTOT	real	array of integrals on tracers	TRACER, STEP	STEP	FULLWD
TTN	real	northward transport of tracers	STEP	STEP	local variable
TTSEC	real	current total elapsed time in seconds	STEP	STEP	FULLWD
TTYEAR	real	number of years of integration completed	STEP	STEP	local variable
TVAR	real	change of variance of tracers	TRACER	TRACER	FULLWD
TX	real	temperature in CSU STATE; temporary array in CSU STRESSUP	STATE, STRESSUP	STATE, STRESSUP	local variable
U	real	zonal component of velocity	STEP	TRACER, STEP, CLINIC	WORKSP
UA	real	similar to U but for timestep after present	CLINIC	CLINIC	WORKSP
UB	real	similar to U but for timestep before present		CLINIC	WORKSP
UBM	real	similar to UB but in row $J-1$		CLINIC	WORKSP
UBP	real	similar to UB but in row $J+1$	STEP, CLINIC	CLINIC	WORKSP
UCLIN	real	array of internal mode component of U at row $J+1$	STEP, CLINIC	CLINIC	WORKSP
UDIF	real	vertical mixing computation array for U	CLINIC	CLINIC	WORKSP
UENG	real	individual forcing terms on U			
UICE	real	ice-drift velocity of three timesteps computed by the ice model	OCEAN	STRESSUP	TSTOP
UICEC	real	immediate ice-drift velocity computed by the ice model	OCEAN	not used	TSTOP
UM	real	similar to U but in row $J-1$		CLINIC, TRACER	WORKSP

DATA ELEMENT	DATA TYPE	DESCRIPTION	CSU DEFINED IN	CSU USED IN	COMMON BLOCK
UOVER	real	positioned in common to be equivalent of UDIF with K index equal to 0; used to set surface boundary conditions	CLINIC	CLINIC	WORKSP
UP	real	U in row $J+1$	STEP, CLINIC	STEP, CLINIC	WORKSP
USAV	real	array of internal mode component of U at row J	CLINIC	CLINIC	WORKSP
UUNDER	real	positioned in common to be equivalent of UDIF with K index equal to $KM+1$	CLINIC	CLINIC	WORKSP
V	real	meridional component of velocity	STEP	TRACER, STEP, CLINIC	WORKSP
VA	real	V at timestep after present	CLINIC	CLINIC	WORKSP
VB	real	V at timestep before present	STEP	CLINIC	WORKSP
VBM	real	similar to VB but in row $J-1$	STEP	CLINIC	WORKSP
VBP	real	similar to VB but in row $J+1$	STEP	CLINIC	WORKSP
VBR	real	zonal average of horizontal velocity	STEP	STEP	local variable
VBRZ	real	zonal/vertical average of horizontal velocity	STEP	STEP	local variable
VCLIN	real	array of internal mode component of V at row $J+1$	CLINIC, STEP	CLINIC	WORKSP
VDIF	real	vertical mixing computation array for V	CLINIC	CLINIC	WORKSP
VENG	real	individual forcing terms on V	CLINIC	CLINIC	local variable
VICE	real	ice-drift velocity of three timesteps computed by the ice model	OCEAN	STRESSUP	TSTOP
VICEC	real	immediate ice-drift velocity computed by the ice model	OCEAN	not used	TSTOP
VM	real	V in row $J-1$		CLINIC	WORKSP
VOLUME	real	total volume of the model basin	OCEAN	STEP	FULLWD

DATA ELEMENT	DATA TYPE	DESCRIPTION	CSU DEFINED IN	CSU USED IN	COMMOI BLOCKS
VOVER	real	positioned in common to be equivalent of VDIF with K index equal 0; used to set surface boundary conditions	CLINIC	CLINIC	WORKSP
VP	real	V in row $J+1$	CLINIC	STEP, CLINIC	WORKSP
VSAV	real	array of internal mode component of V at row J	CLINIC	CLINIC	WORKSP
VUNDER	real	positioned in common to be equivalent of VDIF with K index equal to $KM+1$	OCEAN, STEP	used only in equivalence	WORKSP
W	real	vertical velocity; computed in vertical line with U,V in CLINIC; computed in vertical line with T in TRACER	CLINIC, TRACER	TRACER	WORKSP
WINDSX	real	zonal wind velocity	STRESSUP	STEP, STRESSUP	TSTOP
WINDSY	real	meridional wind velocity	STRESSUP	STEP, STRESSUP	TSTOP
WSX	real	zonal component of surface wind stress	STEP	CLINIC, STEP	WORKSP
WSXM	real	WSX in row $J-1$	not used	not used	WORKSP
WSXP	real	WSX in row $J+1$	STEP	not used	WORKSP
WSY	real	meridional component of surface wind stress	STEP	CLINIC	WORKSP
WSYM	real	WSY in row $J-1$	not used	not used	WORKSP
WSYP	real	WSY in row $J+1$	STEP	not used	WORKSP
YNEG	real	temporary storage used to read negative ice to be melted; heat above the freezing temperature computed by the ice model	OCEAN	TRACER	TSTOP
ZDZ	real	vertical position of bottom of levels	OCEAN	OCEAN	ONEDIM
ZDZZ	real	vertical position of center of levels	OCEAN	OCEAN	ONEDIM
ZTD	real	change of vorticity across one timestep	CLINIC	TRACER	FIELDS

DATA ELEMENT	DATA TYPE	DESCRIPTION	CSU DEFINED IN	CSU USED IN	COMMON BLOCK
ZUN	real	time change of vertically averaged zonal forcing at north face	CLINIC	CLINIC	ONEDIM
ZUNENG	real	vertical average of U forcing at north face	CLINIC	CLINIC	WORKSP
ZUS	real	time change of vertically averaged zonal forcing at south face	CLINIC	CLINIC	ONEDIM
ZUSENG	real	vertical average of V forcing at south face	CLINIC	CLINIC	WORKSP
ZVN	real	time change of vertically averaged meridional forcing at north face	CLINIC	CLINIC	ONEDIM
ZVNENG	real	vertical average of V forcing at north face	CLINIC	CLINIC	WORKSP
ZVS	real	time change of vertically averaged meridional forcing at south face	CLINIC	CLINIC	ONEDIM
ZVSENG	real	vertical average of V forcing at south face	CLINIC	CLINIC	WORKSP

5.0 CSCI PIPS2.0 DATA FILES

This section describes the transfer of data elements between PIPS2.0 CSCs and CSUs via data/ disk files.

5.1 Data File to CSU/CSC Cross-Reference

DATA FILE	UNIT #	READ FROM:	WRITTEN TO:
Levitus Temperature	10	CSU DRIVER	
Levitus Salinity	11	CSU DRIVER	
NOGAPS	12	CSU RFORCE	
River Discharge	19	CSU DRIVER	
Grid Positions	14	CSU DRIVER	
Land/Sea Masks	15	CSU BNDRY	
Ocean Data	13	CSU RST_OCN	
	31		CSU SAV_OCN
Ice Restart Data	16	CSU RST_ICE	
	33		CSU SAV_ICE
Ocean Restart Data	18	CSU ODAM	
	34		CSU ODAM
Graphics Data	31		CSU DRIVER

5.2 CSU ODAM Virtual Disk Access Logical Units

5.2.1 Logical Unit Number KONTRL

Purpose: Logical unit number KONTRL shall contain the timestep counter, total elapsed tim and area and volume of the basin.

Size: The maximum size of the file will be 20 words.

File Access Method: Direct Access (simulated by CSU ODAM).

Data Stored in File (array BIG):

ITT	timestep counter; total number of timesteps completed
TTSEC	total elapsed time in seconds
AREA	area of the ocean basin
VOLUME	volume of the ocean basin
AKNTRL(6)	available for storage of values that need to be saved to the restart file
PAD(10)	used to prevent overwriting on I/O in core-contained mode

5.2.2 Logical Unit Number KFLDS

Purpose: Logical unit number KFLDS shall contain CSCI Ocean two-dimensional horizonta fields.

Size: The maximum size of the file will be 907,200 words.

File Access Method: Direct Access (simulated by CSU ODAM).

Data Stored in File (array BIG):

P	mass transport stream function
PB	mass transport stream function for the previous timestep
ZTD	change of vorticity across one timestep
HR	reciprocals of total depth at U,V points

5.2.3 Logical Unit Numbers LABS(13), LABS(14), LABS(15)

Purpose: Logical unit numbers LABS(13), LABS(14), and LABS(15) contains the primary sla data for the $N-1$, N, and $N+1$ timesteps, respectively.

Size: The maximum size of each file will be 22,320 words.

File Access Method: Direct Access (simulated by CSU ODAM).

Data Stored in File (array BIG):

Both primary slab data and slab incidental data are stored in these files. The primary slab da includes tracer data and U and V components of horizontal velocity. The slab incidental data includ the number of vertical levels of the ocean at T and U,V points and the wind stress data. Prima slab data is three-dimensional with meridional, zonal, and vertical dimensions. Base arrays T, U and V contain primary slab data. Slab incidental data is two-dimensional with meridional and zon dimensions. FKMTP, FKMUP, WSX, and WSY are arrays of slab incidental data. Because of th

Table 5.2.3-1 — Six-Step Cycle Summarizes the Permuting Disk Units Execute

UNIT NUMBER	TIMESTEP NUMBER					
	1	2	3	4	5	6
13	$N-1$, A	$N+1$, B	N, B	$N-1$, B	$N+1$, A	N, A
14	N, B	$N-1$, B	$N+1$, A	N, A	$N-1$, A	$N+1$, B
15	$N+1$, A	N, A	$N-1$, A	$N+1$, B	N, B	$N-1$, B

permuting disk I/O used to handle the data, the meridional or row dimension of the arrays is limited to three.

On timestep 1 and every sixth timestep, logical units 13, 14, and 15 contain the following data:

Logical Unit 13:

TBP	tracer data for $N-1$ timestep for row $J+1$
UBP	zonal component of horizontal velocity for $N-1$ timestep for row $J+1$
VBP	meridional component of horizontal velocity for $N-1$ timestep for row $J+1$
FKMUP	number of vertical levels of ocean at U,V points for row $J+1$
WSYP	meridional component of surface wind stress for row $J+1$

Logical Unit 14:

TP	tracer data for N timestep for row $J+1$
UP	zonal component of horizontal velocity for N timestep for row $J+1$
VP	meridional component of horizontal velocity for N timestep for row $J+1$
FKMTP	number of vertical levels of ocean at U,V points for row $J+1$
WSXP	zonal component of surface wind stress for row $J+1$

Logical Unit 15:

TA	tracer data for $N+1$ timestep
UA	zonal component of horizontal velocity for $N+1$ timestep
VA	meridional component of horizontal velocity for $N+1$ timestep
BCON	used to write out FKMU and WSY on odd timestep, FKMT and WSX on even timestep

Table 5.2.3-1 summarizes the 6-step cycle that the permuting disk units execute. In this table, $N-1$, N, and $N+1$ denote primary slab data for the $N-1$ (timestep before present), N (present), and $N+1$ (timestep after present) timesteps. Slab incidental data is denoted by A and B. FKMU and WSY are A type arrays and FKMT and WSX are B type arrays. On even timesteps, the A and B type data are read into incorrect arrays and must be switched.

6.0 ABBREVIATIONS AND ACRONYMS

CSCI	Computer Software Configuration Item
CSC	Computer Software Component
CSU	Computer Software Unit
NOGAPS	Navy Operational Global Atmospheric Prediction System
NRL	Naval Research Laboratory

ODAM	Ocean Direct Access Manager
PIPS2.0	Polar Ice Prediction System Version 2.0
SDD	Software Design Document

7.0 SUMMARY AND CONCLUSION

Since 1987, FNMOC has been running sea-ice forecasting systems in various regions of Navy interest (Central Arctic, the Barents Sea, and the Greenland Sea). The Polar Ice Prediction Systen (PIPS1.1) predicts sea ice conditions in the Arctic basin, the Barents Sea, and the Greenland Sea at a resolution of 127 km. Two regional sea-ice forecasting systems, the Polar Ice Predictior System–Barents (RPIPS-B) and the Polar Ice Prediction System–Greenland Sea (RPIPS-G), also predict sea ice conditions in the Barents Sea and the Greenland Sea, respectively, at a higher resolution of 20–25 km. In 1995, NRL delivered to FNMOC a coupled ice-ocean system, PIPS2.0 which predicts sea ice conditions for most of the ice-covered regions in the Northern Hemisphere. PIPS2.0 will replace the existing three operational forecast systems when it completes the final operational testing phase at FNMOC. PIPS2.0 uses as its basis the Hibler ice model and the Cox ocean model. PIPS2.0 has a resolution of approximately a quarter of a degree, which is similar to the resolution of the operational regional systems (RPIPS-B and RPIPS-G).

8.0 ACKNOWLEDGMENTS

This work was funded by the U.S. Space and Naval Warfare Systems Command, Program Element 0603207N.

9.0 REFERENCES

Cheng, A. and R. H. Preller, "An Ice-Ocean Coupled Model for the Northern Hemisphere," *Geophysical Research Letters* **19**(9), 901–904 (1992).

Cox, M. D., "A Primitive Equation, 3-Dimensional Model of the Ocean," Geophysical Fluid Dynamics Laboratory Ocean Group Technical Report, Princeton, NJ, 1141 pp., 1984.

Hibler, W. D. III, "A Dynamic Thermodynamic Sea Ice Model," *Journal of Physical Oceanography* **9**, 815–864 (1979).

Hibler, W. D. III, "Modeling a Variable Thickness Sea Ice Cover," *Monthly Weather Review* **10** 1943–1973 (1980).

Maykut, G. A. and N. Untersteiner, "Numerical Prediction of the Thermodynamic Response of Arctic Sea Ice to Environmental Changes," Memo. Rm-6093-PR, Rand Corporation, Santa Monica, CA, 1969.

Parkinson, C. L. and W. M. Washington, "A Large-Scale Numerical Model of Sea Ice," *Journal of Geophysical Research* **84**, 311–337 (1979).

Preller, R. H. and P. G. Posey, "The Polar Ice Prediction System—A Sea Ice Forecasting System," NORDA Rep. 212, Naval Research Laboratory, Stennis Space Center, MS, 1989.

FNMOC INSTRUCTION 5234.5, Fleet Numerical Oceanography Center Software Standards Manual.

Manabe, S., K. Bryan, and M. Spelman, "A Global Ocean-Atmosphere Climate Model with Seasonal Variation for Future Studies of Climate Sensitivity," *Dynamics of Atmosphere and the Ocean* **3**, 393–426 (1979).

Military Standard, Defense System Software Development DOD STD-2167A, Commander, Space and Naval Warfare Systems Command, COMSPAWARSYSCOM-3212, Washington, D.C., 29 Feb 1988.